1981-03

Vehicle routing algorithms for local delivery at Naval supply centers

Clausen, Clifford O.

Monterey, California. Naval Postgraduate School

VEHICLE ROUTING ALGORITHMS FOR LOCAL DELIVERY AT NAVAL SUPPLY CENTERS

Clifford O. Clausen

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

VEHICLE ROUTING ALGORITHMS FOR LOCAL DELIVERY AT NAVAL SUPPLY CENTERS

by

Clifford O. Clausen

March 1981

Thesis Advisor: A.W. McMasters

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
Vehicle Routing Algorithms for Local Delivery at Naval Supply Centers

5. TYPE OF REPORT & PERIOD COVERED
Master's Thesis; March 1981

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Clifford O. Clausen

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93940

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93940

12. REPORT DATE
March 1981

13. NUMBER OF PAGES
131

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Vehicle Routing
Vehicle Scheduling
Local Delivery
Truck Dispatching
Naval Supply Centers

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
This thesis examines the local delivery operations at the Naval Supply Centers in Oakland and San Diego. The local delivery problem is formulated as a model applicable to these supply centers. Specifically, the model involves routing a fleet of vehicles from a central depot to each of a set of customers so as to satisfy their demands. Twelve heuristic solution methods applicable to this model are reviewed and illustrated with examples. They

#20 - ABSTRACT - (CONTINUED)

are also compared with respect to quality of resulting solutions and computational efficiency. Finally, recommendations on improving the routing of vehicles at the two Naval Supply Centers are made.

Approved for public release; distribution unlimited.

Vehicle Routing Algorithms for Local Delivery At Naval Supply Centers

by

Clifford O. Clausen
Captain, United States Army
B.S., United States Military Academy , 1974

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL

March 1981

ABSTRACT

This thesis examines the local delivery operations at the Naval Supply Centers in Oakland and San Diego. The local delivery problem is formulated as a model applicable to these supply centers. Specifically, the model involves routing a fleet of vehicles from a central depot to each of a set of customers so as to satisfy their demands. Twelve heuristic solution methods applicable to this model are reviewed and illustrated with examples. They are also compared with respect to quality of resulting solutions and computational efficiency. Finally, recommendations on improving the routing of vehicles at the two Naval Supply Centers are made.

TABLE OF CONTENTS

I. INTRODUCTION ---------- 7

II. CURRENT LOCAL DELIVERY OPERATIONS ---------- 9

A. BAY AREA LOCAL DELIVERY ---------- 9

B. SAN DIEGO LOCAL DELIVERY ---------- 22

III. THE VEHICLE ROUTING PROBLEM ---------- 34

A. PROBLEM DESCRIPTION ---------- 34

B. SOLUTION METHODS ---------- 36

C. NEAREST NEIGHBOR ALGORITHM ---------- 39

D. CLARK AND WRIGHT ALGORITHM ---------- 41

E. TILLMAN AND COCHRAN ALGORITHM ---------- 45

F. T.J. GASKELL'S ALGORITHM ---------- 48

G. MOLE AND JAMESON ---------- 50

H. HOLMES AND PARKER ---------- 55

I. SWEEP ALGORITHM OF GILLET AND MILLER -------- 57

J. TREE SEARCH ---------- 62

K. A TWO-PHASE ALGORITHM ---------- 71

L. GIANT TOUR ---------- 75

M. R-OPTIMAL ALGORITHM ---------- 79

N. TIME WINDOWS ---------- 82

O. A MANUAL METHOD ---------- 85

IV. A COMPARISON OF ALGORITHMS ---------- 89

A. COMPUTATIONAL EFFICIENCY ---------- 89

B. DATA REQUIREMENTS ---------- 93

V. CONCLUSIONS AND RECOMMENDATIONS ---------- 97

APPENDIX A: THE TRAVELLING SALESMAN PROBLEM -------- 100

A. THE TRAVELLING SALESMAN PROBLEM ---- 100

B. A BRANCH AND BOUND ALGORITHM ------- 101

C. THE R-OPTIMAL HEURISTIC ------------ 111

APPENDIX B: THE MINIMUM SPANNING TREE PROBLEM ------ 116

APPENDIX C: THE SHORTEST PATH PROBLEM ------------- 119

APPENDIX D: AUTOMATED VEHICLE SCHEDULING PROGRAMS -- 123

A. THE IBM VSPX ---------------------- 123

B. IVESS ----------------------------- 125

C. AVS ------------------------------- 126

D. CONCLUSION ------------------------ 127

LIST OF REFERENCES ---------------------------------- 128

INITIAL DISTRIBUTION LIST --------------------------- 130

I. INTRODUCTION

Managers at large Naval Supply Centers have a tremendous need to be able to control, plan and document the distribution of material. Large volumes of material, tight delivery requirements, and a lack of planning data limit the ability of transportation managers to efficiently manage the distribution of material.

To help alleviate these problems, the Naval Supply Systems Command (NAVSUP) is sponsoring the development of the Navy Automated Transportation Documentation System (NAVADS). This system's objectives are to maintain a data base (such as weight and cube data) for support of its various modules, to provide control over mode of shipment, to automate preparation of shipping documentation, to provide a transshipment monitoring and control system and to provide a local delivery scheduling system [Ref. 21].

It is the local delivery scheduling system that is the subject of this thesis. The author has investigated the local delivery operations at the Naval Supply Center (NSC) in Oakland, California and the NSC in San Diego, California for the purpose of selecting a vehicle routing algorithm that can be automated. This investigation led the author to a survey of "vehicle routing problem" (also called the local delivery problem, or the vehicle scheduling problem) solution techniques found in the operations research and industrial

engineering literature. The basic delivery model encompassed in this problem is applicable to the delivery operations at both Oakland and San Diego.

The local delivery operations at Oakland and San Diego are described first to give the reader an understanding of the real local delivery problem. Next the vehicle routing problem model is described and solution techniques are surveyed. Additionally, computational experience that operations researchers have had with these solution techniques is presented. There are several classical and theoretical problems related to the vehicle routing problem. These are discussed in Appendices A through C. The capabilities of a few currently available automated local delivery scheduling systems are described in Appendix D.

II. CURRENT LOCAL DELIVERY OPERATIONS

The local delivery operations at the supply centers in Oakland and San Diego are different in many respects--mostly in the volume of material moved. However, in the case of both San Diego and Oakland, deliveries are made from central depots to customers that are not collocated with the respective supply centers. San Diego makes deliveries from three separate centers of supply all within six miles of each other. However, the items stored at each location are different, so there is no decision required as to which depot delivers to which customer.

The descriptions of the local delivery operations at Oakland and San Diego that follow are somewhat more detailed than is necessary to justify the application of the "vehicle routing problem" model of Chapter III. However, it is felt that an in-depth understanding of these operations is necessary for anyone attempting to implement a Navy-wide automated system for vehicle routing.

A. BAY AREA LOCAL DELIVERY

The Bay Area Local Delivery (BALD) system of the Naval Supply Center at Oakland delivers to customers within approximately 100 miles of the center. According to Hrabosky, Owen, and Popp [Ref. 15: p. 37-41] BALD services 162 shore activities in the Bay Area plus ships in the bay. Recently the cities of Monterey, Stockton, and Tracy as well as Sharpe

supply depot in Sacramento have been delted from BALD runs. Thus BALD makes deliveries to approximately 148 customers. Deliveries are also made to the Military Overseas Terminal Bay Area (MOTBA) in Oakland for further movement by water and to Travis Air Force Base, McClellan Air Force Base, San Francisco Airport and Oakland Airport for further movement by air. The major points of delivery can be clustered into eighteen distinct geographical groupings as listed in Table II-I. Customers are proximately located in these groups, and therefore all customers within each group can be considered as one customer for the purpose of routing vehicles. There are a few customers that do not fall within these clusters. However, the frequency and volume of delivery to these customers does not merit additional groupings (for example, the Naval Reserve Center, San Jose). Bay area customers are depicted on the map in Figure 1.

Currently there are four regularly scheduled routes (called "stakes") that service customers around the bay on a daily basis. The major customers on each stake along with measurement ton data for each stake are given in Table II-II. A measurement ton is a rough estimate of 40 cubic feet. In addition to these four bay area stakes, two trucks per night are sent to Travis A.F.B., one goes to the terminal for logistics support of major Navy Fleet Centers within the continental United States (QUICKTRANS) and the other goes to the Military Airlift Command (MAC) terminal. All material that is to be

TABLE II-I

CLUSTER	LOCATION
1	NSC Oakland
2	Alameda
3	Point Molate
4	Mare Island/Skaggs Island
5	Concord
6	Moffett Field
7	Treasure Island/Yerba Buena Island
8	Hunter's Point
9	San Bruno
10	Presidio, San Francisco
11	Oakland
12	Naval Hospital Oakland
13	San Francisco Piers
14	Travis A.F.B.
15	McClellan A.F.B.
16	Government Island
17	Oakland Airport
18	San Francisco Airport

further transported within the United States and is suitable for shipment by commercial airline goes to the QUICKTRANS terminal. All other air cargo goes to the MAC terminal. A commercial carrier also makes a daily trip from the NSC to the QUICKTRANS terminal. Material destined for the MAC terminal can also be sent on this truck because there is a shuttle that moves material back and forth between the two terminals. Finally, a truck is sent weekly to the terminal for logistic airlift support of Air Force installations (LOGAIR) at McClellan A.F.B.

Material that is to be shipped by BALD is brought into the receiving section of building 341 (BALD warehouse). This material enters the building on pallets, but often is not

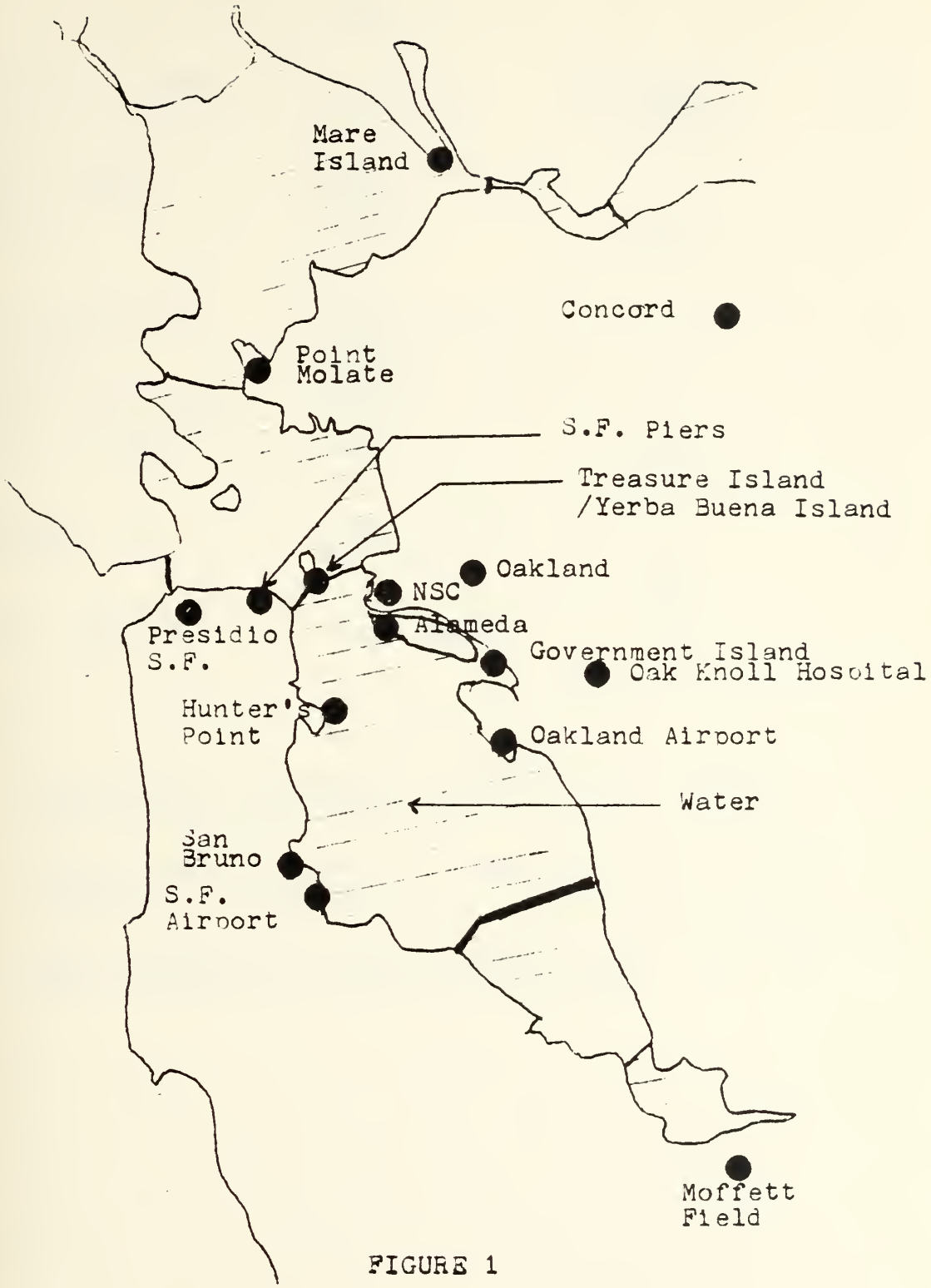

Mare
Island
Concord
Point
Molate
S.F. Piers
Treasure Island
/Yerba Buena Island
Oakland
NSC
Alameda
Presidio
S.F.
Government Island
Oak Knoll Hospital
Hunter's
Point
Oakland Airport
Water
San
Bruno
S.F.
Airport
Moffett
Field

FIGURE 1

TABLE II-II

MAJOR CUSTOMER	FREQUENCY	MEASUREMENT TONS	STAKE
Point Molate	Daily	*	1
Mare Island	Daily	*	1
Concord	Daily	*	1
		22 MT	
Treasure Island	Daily	*	2
S.F. Piers	Thrice Weekly	*	2
Presidio	Weekly	*	2
Hunter's Point	Daily	*	2
NAVFAC, San Bruno	Weekly	*	2
Moffet Field	Daily	*	2
		17 MT	
Alameda	Twice Daily	*	3
Hospital, Oak.	Weekly	*	3
		4 MT	
NARF, Alameda	Twice Daily	4 MT	4

Notes:

1. No individual consignee measurement ton data is maintained by the NSC.

2. Stakes 1 and 2 use a 40 ft van and depart the BALD warehouse at 0800 daily.

3. Stakes 3 and 4 use a $2\frac{1}{2}$ ton stake truck and depart the BALD warehouse at 1000 and 1400 daily.

arranged by customer. Therefore, the material must be taken off the pallets and sorted onto pallets by customer. The material is then staged for delivery in a separate area for each customer on the warehouse floor.

When the drivers from stakes 1 and 2 have completed their daily stake run, they spot their trailers at the BALD warehouse so that they may be loaded for the next day's deliveries. Material on the floor for next-day delivery on stakes 1 or 2 is then loaded into the proper van trailer. In addition, material coming into the warehouse after the trailers are spotted and destined for the next day's delivery is loaded immediately into the appropriate van trailers.

When material comes into the BALD receiving section, the warehouseman removes the shipping documents from the material and takes them to the shipping clerk. The shipping clerk annotates these documents with the date and time that it is expected that the material will be delivered. Material for delivery on stakes 3 and 4 received before 1400 is dated as shipped the same day. Material going to ships is held in the BALD warehouse until the ship arrives in port. All other material is dated for next-day delivery. The annotated shipping document is considered to be proof of shipment of the material and also serves to stop the imaginary clock that tracks the transportation hold time. The actual time the truck departs is not recorded on these documents nor is there a procedure for obtaining proof or time of receipt by the actual customer. No individual customer measurement

ton or weight and cube data are collected either, although the measurement tons on each truck leaving building 341 are recorded.

The trucks that are used by the BALD system for delivery do not belong to NSC Oakland, but are rented on an hourly and mileage basis from the Public Works Center of San Francisco (PWC). PWC also provides the drivers for these trucks. Thus BALD does not run a dispatching operation but merely clusters customers into stakes.

A normal round trip takes about 6.5 to 8 hours on stakes 1 and 2 but takes less than 4 hours on stakes 3 and 4. The drivers on these regular stakes are the same each day. Thus drivers are familiar with where and to whom delivery is to be made. The drivers are told by BALD personnel which customers on their stakes are to receive delivery. The drivers then pick their own routes consistent with the order in which the truck has been loaded. Each driver maintains a trip log for each trip that he makes. In this log the driver records the arrival and departure time for each stop that he makes as well as the odometer reading (see Figure 2).

Trips that are not part of a regularly scheduled run are called "overflow" and are frequently made for the following reasons: material requiring a special truck must be moved (e.g., a ship's propeller); an emergency request must be filled; delivery must be made to a ship. There is no standard procedure for handling overflow material. In the case of

VEHICLE/EQUIPMENT REQUEST AND RECORD
NAVFAC 9-11240/1 (3-68) S/N 0105 LF-004 1160
Supersedes NAVFAC 11240/1 (3-65)

PART A. To be filled in by Requesting Activity

FBH - #341 - NSC	SEMI.	☒ YES ☐ NO
BLDG #341		6739
Dec 17 1980	Stoke #1 Concord & M.I.	
1697011		
BALL		Dec 17 1980

PART B

Dec 17 1980	J. Q.

PART C. DISPATCH AND USE INFORMATION

96 - 24788		06.14-0917
Dec 17, 1980	IN 1600 OUT 0730	8
	METER READING IN 163626 OUT 163537	89
DOE		721

Trip information to be entered on reverse side.

PART D. TRIP INFORMATION

DESTINATION	TIME ARRIVE	TIME DEPART	ODOMETER READING	REPORT TO
1	2	3	4	5
341 W	0152	0322	163539	
312 W	0325		163538	
483 MI	0445		163567	
001			163568	
Maunakea			163571	
380 #11			163572	
527			163573	
289		1300	163574	
Skaggs	1330	1400	163584	
341 W	1500		163627	

I certify equipment was used for OFFICIAL purposes.

1st OPERATOR (Signature)	2nd OPERATOR (Signature)
Dal	

NAVFAC 9-11240/1 (3-68) (BACK)

PWC San Francisco
Driver' Log Form

FIGURE 2

large material (e.g., steel), arrangements are made in advance with PWC to provide a truck to BALD. The truck is loaded at the storage location of the material and moved immediately to its destination. Thus large material never goes through building 341. Material requested by ships is held in the BALD warehouse until the ship reaches port. Thus deliveries made to ships are very large volume shipments. Arrangements are usually made in advance for these deliveries to be made over a two or three day period. These prior arrangements give BALD time to request trailers from PWC so that this material may be preloaded. Emergency requests are handled by requesting a truck from PWC for use as soon as possible.

When BALD requires an overflow truck, it does not contact PWC directly. Instead, it calls the Labor and Equipment Branch which is responsible for all vehicle usage at the NSC. Labor and Equipment in turn calls PWC requesting the vehicle. BALD has no control over the overflow drivers. The major customers receiving overflow deliveries are listed in Table II-III.

The trucks that make deliveries to Travis A.F.B. and McClellan A.F.B. are paid for on the same job order number as the rest of the BALD deliveries. However, the trucks are requested by the packing foreman and not by BALD.

The Public Works Center runs two motor pool and dispatch operations; one is at the supply center at Oakland, and the other is in Alameda. Although PWC is responsible for providing

TABLE II-III

MAJOR CUSTOMER	AVERAGE NUMBER OF DELIVERIES PER WEEK	AVERAGE MT PER DELIVERY
NAS Alameda	14	7
NARF Alameda	14	7
Mare Island	5	10
Alameda Facility	4	24
Ships (NAS Alameda)	4	14
MOTBA	2	14
Ships (Mare Island)	2	14
Hunter's Point	2	12
Ships (Hunter's Point)	1	6
Ships (San Francisco)	1	6

vehicles and transportation to many government activities throughout the San Francisco Bay Area, most trucks are dispatched in support of NSC Oakland. PWC can respond to normal requests for vehicles within one day. Emergency requests for vehicles can be handled within the same day by shuffling dispatches.

The PWC operation is not a budgeted activity, but instead receives its operating funds by charging customers for the support that it renders. The rental rates for trucks and trailers regularly used by BALD and the hourly wage rate for drivers for FY 1981 are listed in Table II-IV. The PWC

TABLE II-IV

COST CODE	DESCRIPTION	MONTHLY	HOURLY	MILEAGE
445	Truck, 2 ton, stake	$152.00	.95	.131
603	Truck, 5 ton, stake	459.00	2.87	.190
620	Truck, Tractor, 10 ton 4X2	816.00	5.10	.390
630	Truck, Tractor, 10 ton 6X4	816.00	5.10	.390
645	Truck, Tractor, 15 ton (diesel powered)	894.00	5.59	.319
816	Semitrailer, 20 ton, stake	101.00	.63	----
817	Semitrailer, 20 ton, van	101.00	.63	----

LABOR: Straight time $22.74/hr
Overtime $28.11/hr

bills its customers for no more than eight hours of trailer use each day which permits the spotting of trailers at no additional charge. They bill for driver and truck usage based upon the actual hours used. Drivers receive overtime for any work in excess of eight hours per day or forty hours per week. Currently, there is no extra charge for emergency requests beyond the possibility of overtime.

The vehicles and drivers used by BALD are paid for by the Naval Supply Systems Command (NAVSUP) in Washington, D.C., on the second destination job order #1687011. Thus, there is no incentive for NSC Oakland or PWC San Francisco, or the customer to be efficient in the use of vehicles. In particular, there are frequent delays in the unloading of trucks. The unloading of trucks is the responsibility of the customer, and drivers sometimes must wait up to half a day to get their trucks unloaded. On some occasions, vehicles requested to make emergency deliveries to the Naval Air Rework Facility (NARF) on Alameda have had to wait because no one at the NARF knew where the material was to be delivered.

Both BALD and PWC are very interested in doing their best to service customers in a timely fashion. In order to meet Uniform Military Movement Issue Priority System (UMMIPS) delivery time frames, standards for transportation hold times have been set by NAVSUP as follows: issue group 1 items have one day; issue group 2, three days; and issue group 3, seven days. The transportation hold time begins once an item has been picked at the warehouse. AS a result the

transportation hold time clock actually starts before the item reaches the BALD warehouse. The clock on an item stops when the shipping clerk annotates the shipping document. The responsibility for moving material from the storage location to the BALD warehouse does not belong to BALD. In fact BALD has no knowledge of shipments until they reach the BALD warehouse.

As a result of this lack of information, there is a delay associated with moving material to BALD. The average transportation hold times in days for the months of July 1980 through November 1980 by issue group are listed below.[1]

	IG1	IG2	IG3
JUL	2.24	1.63	2.88
AUG	2.18	2.31	2.32
SEP	1.83	1.75	2.64
OCT	2.0	2.0	3.05
NOV	1.59	2.23	2.17
NAVSUP STD	1.0	3.0	7.0

Notice that the standard for issue group 1 is violated in every case. This problem cannot be attributed to the BALD section, because all material coming into BALD, regardless of issue group, is delivered on the next scheduled truck. This fact explains why there is little difference in the hold time associated with each issue group.

In an attempt to access the efficiency of the BALD operation, the author examined driver's trip loss, production

---

[1]This data was obtained from BALD production reports.

reports and billing statements against the BALD job order number. Additionally, the author rode with stakes 3 and 4. An examination of 35 trip logs totaling 197 hours of truck usage from the months of November and December 1980 revealed that 27% of the total time was spent servicing the customer, 39% of the time was travel time and 34% of the time included waiting to be loaded, driver breaks and other miscellaneous actions. It was interesting to note that in determining truck usage, PWC usually rounded up to the nearest half hour. Of the records examined, round-up accounted for about 4% of the total time. The average time spent actually servicing the customer was 42 minutes per customer.

The production and cost figures associated with job order #1687011 for the period October 1979 through December 1980 are listed in Table II-V. The measurement ton figures for air cargo have been adjusted downward by 1/3 because approximately 1/3 of the air cargo is sent by commercial carrier and therefore not included in the costs.

In riding on stake 4, it was found that the NARF does not have a central receiving facility. Therefore, the truck driver must make several local stops at different warehouses.

B. SAN DIEGO LOCAL DELIVERY

The Naval Supply Center in San Diego makes local deliveries to about fifty shore activities, including Long Beach, Camp Pendleton, and ships in San Diego harbor. There are three separate NSC facilities. All rackables, binables, chill and

TABLE II-V

Measurement Tons

MONTH	MT AIR	MT BALD	MT TOTAL	TOTAL COST	$/MT
Oct 79	486	8049	8535	54,642	6.40
Nov 79	508	7839	8347	54,079	6.48
Dec 79	592	6403	6995	41,738	5.97
Jan 80	435	8159	8594	52,852	6.15
Feb 80	577	10122	10699	55,410	5.18
Mar 80	574	9174	9748	49,398	5.07
Apr 80	491	10005	10496	52,787	5.03
May 80	867	8666	9533	45,283	4.75
Jun 80	550	8741	9291	42,591	4.58
Jul 80	719	9441	10160	52,712	5.19
Aug 80	571	7986	8557	30,786	3.60
Sep 80	760	8645	9405	44,361	4.72
Oct 80	820	9574	10394	38,404	3.69
Nov 80	574	6327	6901	44,278	6.42
Dec 80	505	5722	6227	37,492	6.02

frozen items are stored at the Harbor Boulevard facility (also the administrative headquarters for the NSC). California Ice, a commercial firm, stores all fresh fruit and vegetables and is located on Imperial Street about two miles south of the Harbor Boulevard facility. All other material, including dry provisions, are stored at the National City Annex (NCA) which is about six miles south of the Harbor Boulevard facility. There are two other NSC facilities in the San Diego area, the NSC fuel division is located at Point Loma and the Naval Air Station annex on North Island. Neither of these facilities are involved in local delivery.

All material except frozen or chill stored at the Harbor Boulevard facility and destined for ships is consolidated at Building 12 on the naval pier across the street from the main facility. This material is then moved to the NCA for subsequent delivery. Food items destined for ships (from either Harbor Boulevard or California Ice) are trucked directly. All shore activities receive delivery directly from the facility where the items are stored.

Except for aircraft carriers, ships dock at the National City piers. The carriers dock at the NAS on North Island. The NCA makes deliveries to ships at the National City piers with straddle trucks. Straddle trucks are vehicles specifically designed to carry palletized material short distances (e.g., between warehouses). A straddle truck only services one ship before returning to the annex for another pick-up. The NCA also provides fork lifts to the ships to aid in unloading trucks from either Harbor Boulevard or California Ice. All shipments to the carriers on North Island are sent by truck.

The transportation hold times by issue group are listed below.[2]

The standards for transportation hold time are the same as those for Oakland. Notice that as with BALD, there is a problem in meeting these standards, although in this case, it is issue group 3 instead of issue group 1 that presents

---

[2]This data was obtained from monthly production reports of the local delivery section, NSC San Diego.

MONTH	IG1	IG2	IG3
Apr 80	1.55	4.52	7.19
May 80	.84	2.44	8.81
Jun 80	1.05	4.08	9.87
Jul 80	----------NOT AVAILABLE----		
Aug 80	.59	2.58	9.87
Sep 80	1.22	4.34	9.14
Oct 80	1.21	5.03	9.09
Nov 80	.85	2.74	5.98
Dec 80	1.13	2.20	10.78
Jan 81	1.22	3.0	7.82
NAVSUP STD	1.0	3.0	7.0

the problem. Currently, the choice of what material to deliver is made as material enters the delivery section. As with BALD, no advance knowledge of delivery requirements is available to the delivery section. Thus local delivery is planned on a day to day basis.

The trucks used by the NSC to make deliveries are rented from PWC San Diego which is located in National City. Unlike Oakland, trucks are rented on a monthly basis and the drivers are navy personnel assigned to the NSC. The NSC is responsible for fueling the trucks and for scheduling maintenance. However, actual maintenance work is performed by PWC.

Deliveries of chill, frozen, and fresh food items are most frequently made by 40 ft. refrigerated vans, although smaller trucks are available. Delivery of other material is most frequently made with 40 ft flat-bed trucks. Other types of trucks may also be used. A list of trucks used for

local delivery along with rental rates is displayed in Table II-VI.

Each truck is equipped with a two-way radio capable of contacting the dispatcher located at Harbor Boulevard. Drivers contact the dispatcher upon arriving at and departure from customers and report any problems in making delivery. Drivers also fill out a log that lists location, arrival time, departure time, and the number of pallets hauled. A sample log is shown in Figure 3.

The San Diego local delivery system delivers about 35,000 pallets of material per month via navy carriers. An additional 5000 pallets per month are shipped through a commercial carrier. The numbers of pallets delivered each month from October 1979 to January 1981 are listed in Table II-VII. Notice that San Diego reports production in pallets and Oakland in measurement tons. These two units are approximately the same in that both are rough estimates of 40 cubic feet.

Both the commercial and navy carriers make deliveries to Long Beach. Long Beach, an annex of the NSC in San Diego, serves ships and facilities in the Long Beach area. The average transport time to Long Beach by commercial carrier is about three days whereas the navy carrier transport time is only one day.

Local customers are divided into zones as shown in Figure 4 so that they know when to expect deliveries of material. Zone deliveries are made according to the following schedule:

TABLE II-VI

CODE	DESCRIPTION	FY81 Hourly	FY81 Mileage	FY81 Man
H 0308	Truck, 1/2 T, Utility	1.08	.15	173
G 0313	Truck, 1/2 T, Pickup	1.03	.15	165
G 0316	Truck, 1/2 T, Pickup	1.03	.15	165
H 0327	Truck, 3/4 T, Pickup	1.08	.15	173
H 0330	Van, Carryall	1.08	.15	173
F 0342	Truck, 1T, Pickup	1.08	.16	173
I 0345	Truck, 1T, Step	1.08	.16	173
J 0443	Truck, 2T, Dump	1.03	.21	165
J 0445	Truck, 2T, Stake	1.03	.21	165
M 0590	Truck, 5T, Van	1.54	.21	246
M 0603	Truck, 5T, Stake	1.54	.21	246
M 0604	Truck, 5T, Truck Tree	1.54	.21	246
M 0605	Truck, 5T, Van	1.54	.21	246
M 0614	Truck, 7-$\frac{1}{2}$ T, Truck Tree	1.69	.16	270
M 0617	Truck, 10T, Trk/Tractor	1.69	.16	270
M 0633	Truck, 10T, Trk/Trac 6x4 Diesel Powered	1.69	.16	270
N 0645	Truck, 15T, Truck/Trac	2.67	.26	427
P 0817	Trailer, 20T, Van	.46	---	74
P 0820	Trailer, 20T, Van Refrig	.46	---	74
P 0827	Trailer, 51-60T,	.46	---	74
P 0860	Trailer, Tank, 400 Gal	.10		16
P 0890	Trailer, Tank 4-6K Gal	.46		74

LOCAL DELIVERY INDIVIDUAL PRODUCTION REPORT
NSC-SD-5'00 70 (10-80) 96-37584 - 97-22845

DRIVER

TYPE VEHICLE	TYPE MATERIAL	DESTINATION	ZONE	ARRIVAL TIME	DEPARTURE TIME	MATERIAL HAULED CUSTOMER	MATERIAL HAULED IN HOUSE	TRANSIT TIME	BROADWAY	NCA	MISC	REMARKS
K	ERZ	BLDG. 7 NSC		0730	0756							
	"	SUB BASE GALLEY		0822	0840	5						
	"	SPERRY		0846	0930	25						
		FUEL DEPT PLA		0936	0944							
		NSC COMPOND		1000	1015							
		CAL-TRE		1025	1100							
	CHL	SAN CLEMENTE		1117	1122	1						
	1'	KITTY HAWK		1125	1224	15						
		GARAGE		1244	1333							DROPPED TRAILER
A	97-31354	NCA COMPOND		1337	1347							PICKED UP FLT BED
		BLDG 279		1355	1425							
		BLDG. 65		1430	1437							
		BLDG 11 NSC		1454	1502		1					
		NOSC		1520	1535	13						
		NSC		1554								
		STRADDLE TRUCK		1600	1700		25					

TYPE VEHICLE
A - 40' FLAT
B - 32' FLAT
C - 7.5T STAKE
D - 5T STAKE
E - 2.5T STAKE
F - 1ST P/U
G - .75T P/U
H - 40' VAN
I - 36' VAN
J - 5T VAN
K - 40' REEFER
L - 35' REEFER
M - 30' REEFER
N - 5' REEFER
O - STRADDLE TRUCK

TYPE MATERIAL
GSK - GENERAL SUPPLIES
STL - STEEL
DRY - DRY PROVISIONS
FRZ - FREEZE
CHL - CHILL
PLT - PALLET

REMARKS
A - NO FORK LIFT
B - NO AMPY PARTY
C - NO CRANE
D - PIER BLOCKED
E - PIER CLOSED
F - REJECTION
G - MECHANICAL PROBLEMS

San Diego Driver's Log Form

* TO BE FILLED IN BY DRIVER

*TOTAL HAULED 59 26

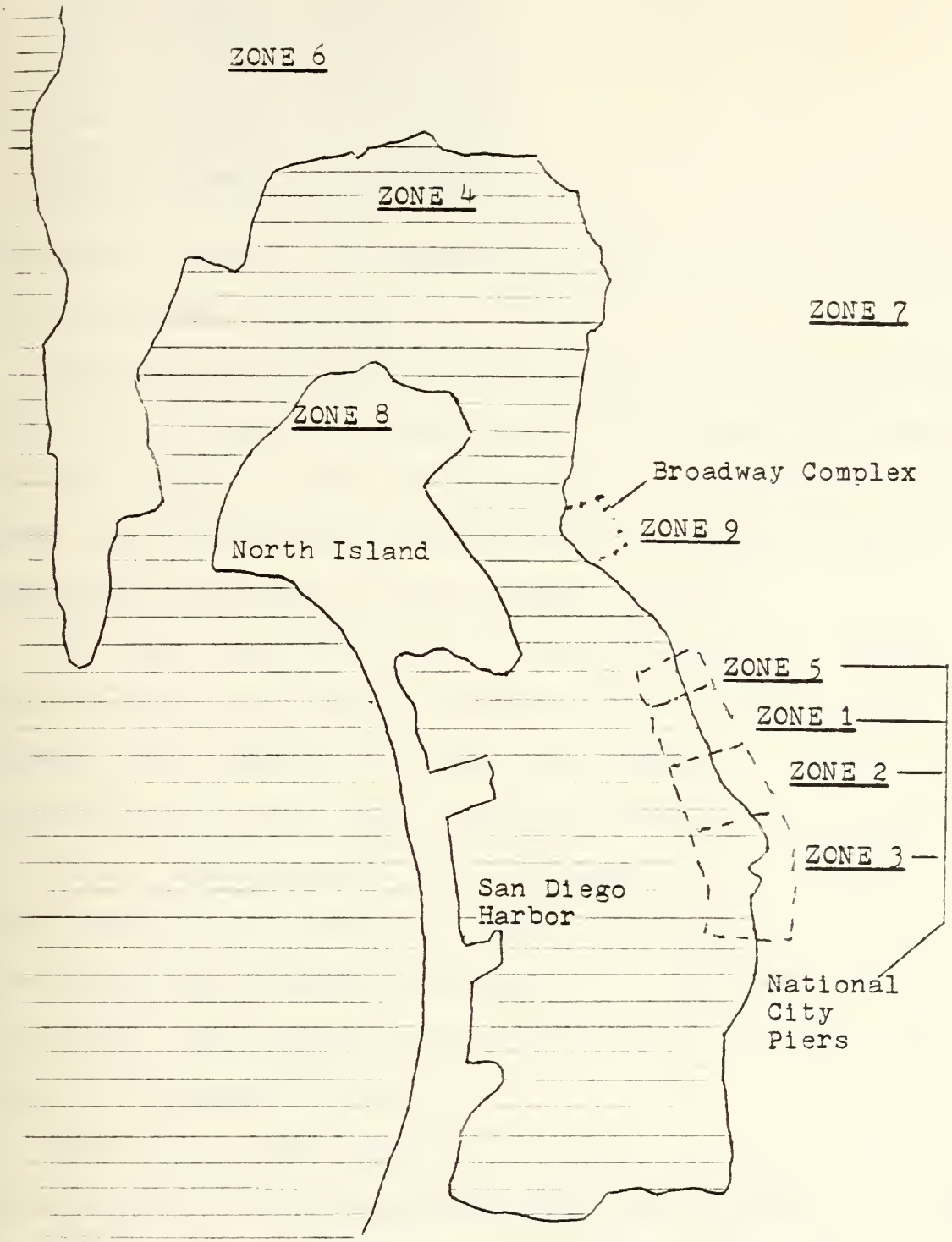
*DATE. 1/27/81

FIGURE 3

TABLE II-VII

Total Pallets of Material Moved by N9C Trucks

MONTH	PALLETS
Oct 79	32,136
Nov 79	31,184
Dec 79	29,081
Jan 80	38,166
Feb 80	33,811
Mar 80	35,013
Apr 80	37,455
May 80	35,608
Jun 80	36,788
Jul 80	39,846
Aug 80	34,631
Sep 80	35,064
Oct 80	37,940
Nov 80	28,491
Dec 80	36,769
Jan 81	38,625



San Diego Zone Map

FIGURE 4

Zone 1--Monday and Thursday
Zone 2--Monday and Thursday
Zone 3--Tuesday and Friday
Zone 4--Monday through Friday
Zone 5--Monday and Wednesday
Zone 6--Tuesday and Thursday
Zone 7--Monday and Wednesday
Zone 8--Tuesday and Friday
Zone 9--Monday through Friday

Provisions, however, are delivered anywhere on any working day regardless of the zone.

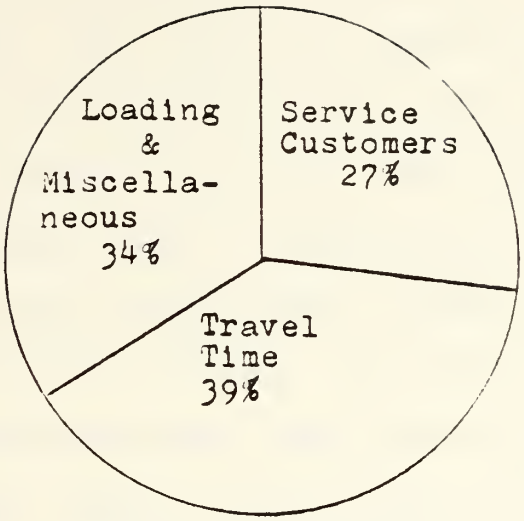
Shore activities can generally accept delivery any time during the working day. Ships usually can accept delivery any time during the day unless scheduled to leave port, in which case delivery must be made before a specified time. They prefer to receive in the morning, however.

The costs associated with the local deliveries around San Diego were not obtained in this work. However, twenty drivers' logs totaling 126 hours were examined. It was found that 33% of the total time was spent unloading at customers, 31% of the time was spent travelling and 36% of the time was spent loading at the depot and other miscellaneous actions. Furthermore, the average time spent at a customer's site was 27 minutes. The Oakland and San Diego local delivery operations are compared in Figure 5.

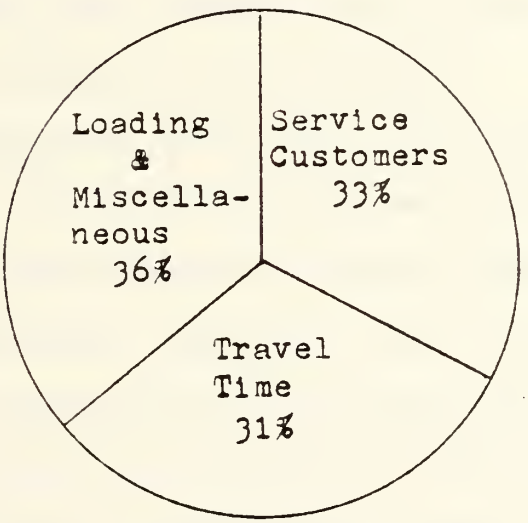
The NSC has tried scheduling vehicles using the IBM Vehicle Scheduling Program Extended (VSPX--see Appendix D). VSPX is reported to have produced good feasible routes and to have utilized truck capacity well.[3] The unit of capacity

---

[3]Marlo Olsen, interview at NSC San Diego, February 4, 1980.



OAKLAND LOCAL DELIVERY



SAN DIEGO LOCAL DELIVERY

FIGURE 5

used was the pallet which also worked quite well. The major problem leading to the discontinued use of VSPX was the daily data collection effort required. The number of pallets by customer had to be manually transfered to cards before VSPX could be run. Data had to be hand-collected because there was no way of relating the line item number of the request with the portion of a pallet that the item would occupy. In order to complete vehicle schedules by 0630, it was necessary to deliver the hand collected data to data processing by 1330 the previous day. However, the delivery section worked until 1600 and did not really know the next day's delivery requirements until the end of the day. Another problem with VSPX was the standardization required to implement the package. Some drivers would carry pallets that were double stacked while others would not. Thus it was difficult to enter the capacity of trucks into the program.

The data collection required to initialize the system was tremendous. The distances between each pair of customers had to be collected because the exact distance option of the VSPX package was used. Moreover, time standards for off-loading at each customer had to be developed. Also, VSPX did not provide any data collection capability that could be used to evaluate the efficiency of the delivery system. However, in spite of all of these difficulties, VSPX would have been a valuable tool if the daily data collection problem could have been solved.

III. THE VEHICLE ROUTING PROBLEM

The classical vehicle routing problem (VRP) analyzed in the operations research literature involves routing a fleet of vehicles from a central depot so as to service a set of customers with known demand and so as to minimize some distribution objective. In this chapter the VRP is described as it relates to local delivery operations at NSCs and solution techniques are surveyed.

A. PROBLEM DESCRIPTION [Ref. 3: p. 315-317]

Consider a time period T over which deliveries must be made to a set N of customers. There is a set V of vehicles available to make deliveries during the time period T. The ith customer is characterized by the following:

1. A demand of Q(i) units;
2. A time required to unload (this may depend on truck type);
3. An early time and a late time forming a time window during which deliveries can be made.

The jth vehicle may be characterized as follows:

1. A total capacity L(j) measured in the same units as demand;
2. An early time and a late time forming a period during which the vehicle can operate;
3. A time required to load at the depot.

All customers and the depot are interconnected by a road network such that the distance between any customer and any other customer (or the depot) is the shortest possible distance. The road system, customers and depot, collectively, may be thought of as forming a network. The set N of customers and the depot are represented by nodes. The shortest path between two customers (or the depot and a customer) along the road system corresponds to an arc connecting the related nodes. The length of the shortest path along the road network is set equal to the length of an arc. In many cases, the time it takes to travel between two points is of more interest than the distance between the two points. In such cases, the length of an arc may be throught of as corresponding to a travel time instead of a distance.

A route, in the sense of the VRP, is a path starting at node P(0), which corresponds to the depot, passing once through each node in a subset of all the nodes and ending at P(0). The vehicle routing problem is to find a set of routes that together pass through every customer, satisfy a set of conditions and minimize some objective. There are many conditions that may have to be satisfied. Conditions applicable to NSC local delivery are:

1. The total demand of all customers on a route may not exceed the capacity of the vehicle assigned to that route;
2. Trucks may only operate during specified hours each day;

3. Delivery to customer P(i), at node P(i), should be made within that customer's time window.

The time window restriction occurs infrequently. Shore activities can almost always accept delivery whenever it arrives and a ship can usually accept delivery as long as they know when it is coming.

There are many possible objectives to the VRP. Two possibilities are to minimize the number of vehicles required to make delivery and another is to minimize the total vehicle time (distance) used to make delivery. The first objective mentioned is equivalent to minimizing fixed costs and the second to minimizing variable costs. However, many non-optimal solution methods (heuristics) cannot distinguish very well between these two objectives [Ref. 3: p. 327].

B. SOLUTION METHODS

Many algorithms have been developed in an attempt to solve the vehicle routing problem. These algorithms fall into two general categories--exact algorithms and heuristic algorithms. Exact methods, when carried through to completion, are guaranteed to find the optimal solution. The problem with exact methods is that, as the number of customers increases, the computation time involved in solving the problem increases and quickly becomes prohibitive [Ref. 11: p. 122]. An exact method for solving the VRP is briefly described in Section J of this chapter.

Heuristic algorithms are not guaranteed to produce optimal or even near-optimal solutions. The advantage of heuristics is that a solution can be obtained in a short time relative to exact algorithms. In addition, a good heuristic will usually provide optimal or near-optimal solutions to large problems. Moreover, the constraints of day-to-day operations can be easily included in the logic of these algorithms [Ref. 2]. For these reasons, this chapter concentrates on heuristic techniques.

The two fundamental steps of all heuristic techniques are grouping customers and sequencing customers. How this is done and the order in which it is done varies from method to method. Some methods sequence before grouping, while others group and then sequence. Still other methods perform both steps at the same time. Heuristic algorithms can also be grouped into the categories of sequential route builders and multiple route builders. Sequential route building algorithms build one route at a time while multiple route building algorithms build more than one route at a time.

Generally, algorithms which build routes sequentially run faster and require less computer memory because fewer feasibility checks need to be made, and once a route is built it can be output rather than kept in memory. Multiple-route algorithms usually, though not always, give better solutions, but take longer to run and require more computer memory.

The sequencing of customers, in many algorithms, requires the solving of the travelling salesman problem (TSP). The

TSP involves routing a single vehicle from a depot through every customer in a set of customers and back to the depot so as to minimize distance (or time or cost). The TSP is discussed in Appendix A. Other problems that arise in the discussion of the VRP are the minimum spanning tree problem (Appendix B) and the shortest path problem (Appendix C).

The data requirements for all algorithms discussed in this chapter are the same, with the exception of the sweep algorithm. The additional data requirements for the sweep algorithm are discussed with that algorithm in Section I. The data needed for the basic algorithms are number of available vehicles by type, vehicle capacity by type, customer demand, and a matrix of shortest distances between stops.

In the sections which follow, algorithms are described which have applicability to the routing of vehicles for local delivery at Naval Supply Centers. All include vehicle capacity and route time constraints. An algorithm which includes time window conditions is discussed in Section N. In all of the algorithms that follow, it is assumed that the demand of each customer is less than the capacity of a vehicle. In cases where this is not true, full vehicles are routed to each customer whose demand exceeds the capacity of one vehicle until the remaining demand for each customer is less than one vehicle load. Furthermore, the distance D(i,j) between two customers P(i) and P(j) is assumed to be symmetric. In other words D(i,j) is equal to D(j,i) for all customers P(i) and P(j).

The chapter concludes with a manual procedure for routing vehicles. Solution quality and computation time are discussed in Chapter IV.

C. NEAREST NEIGHBOR ALGORITHM

The "nearest neighbor" algorithm is a heuristic first developed by M.S. Tyagi [Ref. 25]. In his original explanation of the algorithm, Tyagi did not consider the possibility of a distance constraint. This constraint can be added, but some modification to the method is required. In the explanation that follows, Tyagi's algorithm is modified to include the distance constraint.

The general method of the algorithm is to sequentially build routes (i.e., build one route at a time) by first grouping customers into routes and then sequencing the customers within these routes. Customers are grouped by using the nearest neighbor method. Starting at the dept, P(0), find the nearest customer, say P(1), and add this customer to the first route. At this point, this route goes from P(0) to P(1) and back to P(0). Record the distance travelled, 2D(0,1), and the total demand, Q(1), for this route. Next find the unrouted customer, say P(2), that is closest to P(1). Calculate the total route demand Q(1) +Q(2) and total route distance D(0,1) +D(1,2) +D(2,0) that would occur if P(2) were added to the route. If neither capacity nor distance constraints are violated, P(2) is added to the route and the process of finding the nearest customer to the last feasibly

added customer continues. Total route demand when customer P(m) is added to the route is Q(1) + Q(2) + ... + Q(m) and total route distance is D(0,1) + ... + D(m-1,m) + D(m,0) where P(m) is the last feasible customer added to the route.

If the first route constraint to be violated is the capacity constraint, a modification to the above procedure is made in an attempt to increase vehicle utilization. If customer P(m) is the last customer added to a route and customer P(m+1) causes the route capacity to be exceeded, then customer P(m+1) is considered for possibly replacing either customer P(1) or P(m). The replacement, if either, which results in the largest increase in vehicle utilization is made. At this point the current route is closed from further additions of customers and a new route is formed.

If at any stage the distance constraint is violated, it is possible that further customers may still be added to the current route because the customer sequence within the route is not optimal. Thus, customers may be resequenced using a travelling salesman algorithm before concluding that no further customers can be added. This resequencing is accomplished by temporarily adding to the route the customer P(m+1) that violated the distance constraint and solving the travelling salesman problem (TSP). If the result is a distance-feasible route, the process of adding the nearest neighbor to the last customer in the resequenced route continues until another constraint is violated. If the route still is not feasible, customer P(m+1) is removed from the

route and the route closed from further additions of customers. Once all routes have been formed, each one is resequenced using a travelling salesman algorithm. Customers are resequenced within routes to insure that the travel distance of each route is a minimum.

D. CLARK AND WRIGHT ALGORITHM

Clark and Wright [Ref. 4] first developed the concept of routing vehicles based on the combining of customers into routes by maximizing savings. This method attempts to allocate vehicles to customers such that all customer demands are satisfied and the total mileage traveled is a minimum. Although Clark and Wright in their original formulation did not include the possibility of a maximum allowable distance (or time) for each route, it is easily included into the method and will be described here as part of the method [Ref. 23].

The idea is to initially allocate one truck to each customer so that all customer demands are satisfied. Therefore, there must theoretically be as many trucks as there are customers. Realistically, this is not going to be the case. However, it can be assumed initially that there are enough trucks to make this allocation of trucks to customers--later when customers have been combined into routes, there will probably be enough trucks to service all of the routes formed.

This method makes sequential choices of which customers to combine into routes based upon a largest savings criterion. Initially each customer is linked directly to the depot. Thus, if there are n customers, there are n initial routes. If two customers, P(i) and P(j), are then combined into one route, the total travel distance is reduced by an amount $S(i,j) = D(0,i) + D(0,j) - D(i,j)$ where S(i,j) represents the savings resulting from combining customers P(i) and P(j). This reduction in travel distance results from the fact that initially, the total distance travelled to service customers P(i) and P(j) is: $2D(0,i) + 2D(0,j)$. When customers P(i) and P(j) are linked into one route, the distance travelled to service these two customers becomes $D(0,i) + D(0,j) + D(i,j)$. The savings S(i,j) is the difference between the original travel distance and the new travel distance.

An example inter-distance matrix for a five customer problem is shown in Table III-I. The savings S(1,2) obtained by linking customers P(1) and P(2) into one route is $D(0,1) + D(0,2) - D(1,2) = 10 + 12 - 3 = 19$. The ordered list of savings associated with each link P(i) - P(j) are given in Table III-II.

Once the savings associated with each link P(i) - P(j) have been calculated, the algorithm proceeds to add links starting with the link which has the highest associated savings and which can be feasibly added. Before a link can be added, the following feasibility checks must be made.

TABLE III-I

Q	P(D)					
1500	10	P(1)				
400	12	3	P(2)			
400	8	7	4	P(3)		
400	6	13	13	8	P(4)	
400	5.5	5	12	10	11	P(5)

TABLE III-II

LINK	SAVINGS
P(1)-P(2)	19
P(2)-P(3)	16
P(1)-P(3)	11
P(1)-P(5)	10.5
P(3)-P(4)	6
P(2)-P(5)	5.5
P(2)-P(4)	5
P(3)-P(5)	3.5
P(1)-P(4)	3
P(4)-P(5)	.5

1. The time taken to service the newly formed route may not exceed the allowable working time of a vehicle.

2. There must be a vehicle available with sufficient capacity to service the combined demand of all customers on the new route.

3. The two customers being linked together may not already be on the same route.

4. Both of the customers being linked together must be end points (i.e., first or last customer) on a route. This restriction prevents a customer from being linked to more than two other customers.

Once a link has been added between two customers, it is never subsequently removed.

As an illustration of this algorithm, consider the inter-distance matrix of Table III-I. Furthermore suppose that each customer P(i) has demand Q(i) given in column 1 of Table III-I. Two trucks are available--one with capacity 1200 and the other with capacity 1950. The highest savings is associated with link P(1)-P(2). A truck is available with sufficient capacity to service the combined demand of customers P(1) and P(2) and all of the other feasibility criteria are satisfied so the link P(1)-P(2) is added. At this point there are four routes as follows: P(0)-P(1)-P(2)-P(0); P(0)-P(3)-P(0); P(0)-P(4)-P(0); and P(0)-P(5)-P(0).

The next link from Table III-II is link P(2)-P(3). If this link were added, a new route P(0)-P(1)-P(2)-P(3)-P(0) would be formed with a total demand of 2300. However, there

is not a vehicle with sufficient capacity to service a route with a demand of 2300. Thus the link P(2)-P(3) cannot be added and the first route is complete. To establish the first linking on the second route, Table III-II is examined again, excluding all links involving customers on the first route. Those remaining are P(3)-P(4), P(3)-P(5), and P(4)-P(5). The largest savings occurs for P(3)-P(4). Then link P(3)-P(5) is added and the second route is complete. In summary, the two routes are P(0)-P(1)-P(2)-P(0) with a route distance of 25 and P(0)-P(5)-P(3)-P(4)-P(0) with a route distance of 29.5. The combined distance of the two routes is 54.5.

E. TILLMAN AND COCHRAN ALGORITHM

A simple extension of the Clark and Wright algorithm is that of Tillman and Cochran [Ref. 24]. This method results in solutions that are better than those of Clark and Wright but at the expense of an increase in computation time. This method differs only in the selection criterion that is used to link customers into routes. The calculation of savings and all other aspects of the method are exactly the same.

In the Clark and Wright algorithm the criterion for choosing a link is greatest savings. In this revised method the criterion is to select the best link that allows a second link to be chosen such that the combined savings of the two links is greatest. The example of Section D serves to illustrate this new link selection criterion. In Table III-II, the

highest savings obtainable by a single link is 19 when customers P(1) and P(2) are linked. Once P(1) and P(2) are combined into the same route, no additional customers can be put into this route because the capacity constraint of the largest available truck (here 1950) would be violated.

The Cochran and Tillman method looks beyond the highest feasible savings to investigate the highest feasible savings obtainable by adding two feasible links (see Table III-III). It does this by temporarily adding the feasible link yielding the highest possible savings; this is again 19 for the link P(1)-P(2). It then adds the feasible link yielding the second highest savings; this is the link P(3)-P(4). Notice that links P(2)-P(3), P(1)-P(3), and P(1)-P(5) all have higher savings associated with them than does link P(3)-P(4). However, due to capacity constraints, these three links are not feasible (remember the truck capacities are 1950 and 1200). Then the total savings obtained from both links is recorded. It is 25 for this combination of links. The list of savings is next examined for another possible combination of two

TABLE III-III

LINK 1	1-2	2-3	1-3	1-5	3-4	2-5	2-4	3-5	1-4	4-5
SAVINGS	19	16	11	10.5	6	5.5	5	3.5	3	.5
LINK 2	3-4	1-5	2-5	2-3	1-2	2-3	1-2	1-2	2-3	1-2
SAVINGS	6	10.5	5.5	16	19	16	19	19	16	19
TOTAL SAVINGS	25	26.5	16.5	26.5	25	21.5	24	22.5	19	19.5

links. This time it is based on the feasible link yielding the second highest savings. A search is then made for a feasible link that, when added, yields a total savings of the two temporary links that is greatest. In this case the links P(2)-P(3) and P(1)-P(5) have been added yielding a total savings of 26.5. This process is continued by looking at the possible combinations of two links by considering the third, fourth, and so on highest feasible links and temporarily adding to each of these links the feasible link yielding the highest savings. The total potential savings for each combination is recorded. The process ends when all links have been tried as the first link.

After the above search has been made, the highest total savings is selected and the first link of the two links yielding the highest total savings is added permanently. In this case there is a tie between link P(1)-P(5) and P(2)-P(3) as the first link to be added. Hallberg and Kriebel [Ref. 12] suggest breaking ties by selecting the link with the shortest length. Thus, in the example problem, the first link to be permanently added using the Tillman and Cochran method is link P(2)-P(3). The method carried to its conclusion on the example problem yields routes P(0)-P(1)-P(5)-P(0) and P(0)-P(2)-P(3)-P(4)-P(0) for a total distance travelled on both routes of 50.5. Thus, on this problem the Tillman and Cochran algorithm produces a better solution than the Clark and Wright method.

F. T.J. GASKELL'S ALGORITHM

Gaskell [Ref. 9] suggested three extensions of the basic Clark and Wright algorithm. Two of these extensions differ in the way that the savings are calculated and the third differs in the logic used to form routes. Additionally, he suggested a generalization of the savings calculation that can be used to produce several routes that may be compared.

The savings calculation of Clark and Wright tends to emphasize combinations of customers that are furthest from the depot. This is true because savings result by dropping links between the depot and customers. Thus, the further from the depot the customers are, the greater the savings is likely to be. Gaskell suggested that this method of calculating savings places too much emphasis on the distance of customers from the depot and not enough emphasis on the mutual proximity of customers. The result is that a route restricted by load tends to have most of its customers far away from the depot (peripheral routes) while a route restricted by distance tends to have more customers close to the depot.

To reduce the emphasis placed on the distance from the depot, Gaskell suggested two new savings calculations.

a) $LB(i,j) = S(i,j)(DA + D(0,i) - D(0,j) - D(i,j))$

b) $PI(i,j) = S(i,j) - D(i,j)$

where $LB(i,j)$ is the first modified savings formula; $PI(i,j)$ is the second modified formula; DA is the average of all the distances between the depot and each customer; all other terms

are the same as those used in the explanation of the Clark and Wright algorithm (see reference 9 for a more detailed justification of these formulas). Both LB(i,j) and PI(i,j) give a higher priority to points that are closer to the depot than does S(i,j). The logic used to combine customers into routes is the same as that of the Clark and Wright algorithm.

The third method suggested by Gaskell differs in how the routes are formed. The savings calculation, however, does not differ from that of Clark and Wright. In the Clark and Wright algorithm all routes are considered at the same time (multiple routes). In this method routes are formed one at a time (sequentially). Thus, the only links that can be made are links to end points of the route currently under construction. When there are no more feasible links that can be added to the current route, it is output and a new route is formed. The results of this method are generally inferior (though not always) to multiple routing methods.

The PI savings formula was generalized by Gaskell into the following formula:

$$M(i,j) = D(0,i) + D(0,j) - (TH)D(i,j)$$

where M(i,j) is the generalized savings; TH is a parameter that may vary from problem to problem; all other variables are as before. Levy, Golden and Assad [Ref. 17] have suggested solving the problem with several different values of TH and then picking the best resulting solutions.

G. MOLE AND JAMESON

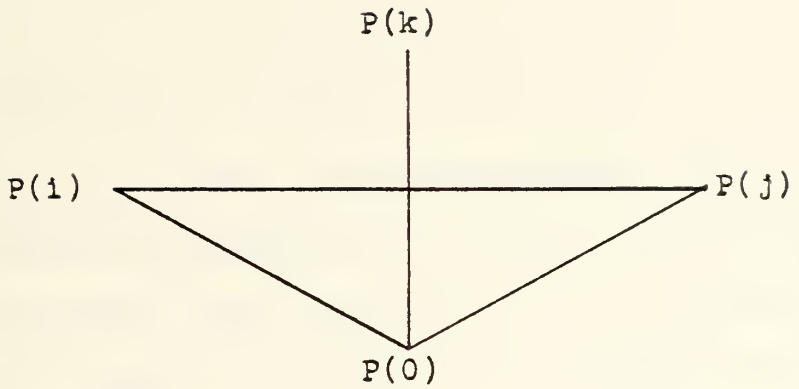
The method of Mole and Jameson [Ref. 19] is a sequential route building algorithm using a savings criteria based on a generalization of the Clark and Wright savings formula. The algorithm consists of three steps that are repeated over and over again until all customers have been routed or until it is not feasible to route any more customers. An optional fourth step then refines the results of the first three steps by looking for ways to move customers between routes.

Step one looks for the best place in the route currently under construction to place each customer that has not yet been routed. Step two then selects from all the unrouted customers the one that is best placed in the current route. The third step is a travelling salesman algorithm that resequences the customers in the current route. The travelling salesman algorithm used by Mole and Jameson was the 2-optimal heuristic method of Lin [Ref. 8], although any reasonable algorithm can be used (see Appendix A).

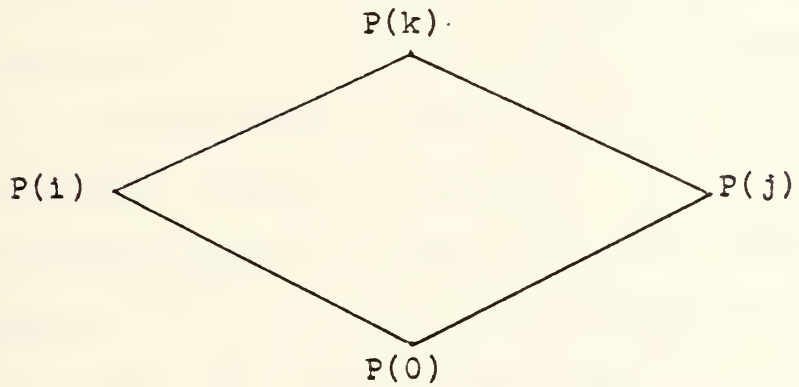
Steps one, two and four use the generalized savings criteria of Mole and Jameson to make decisions. The savings that result from inserting unrouted customer P(k) between routed customers P(i) and P(j) is given by:

$$SV(i,k,j) = 2D(0,k) + D(i,j) - D(i,k) - D(j,k).$$

The logic for this savings formula is illustrated in Figure 6. In Figure 6(a) there are two routes with a combined total distance of $2D(0,k) + D(0,i) + D(i,j) + D(0,j)$. In Figure 6(b)



(a)



(b)

FIGURE 6

there is one route with a total distance of D(0,i) + D(i,k) + D(k,j) + D(0,j). The difference between these two distances is SV(i,k,j). This formula can be rewritten in the form:

$$SV(i,k,j) = 2D(0,k) - ST(i,k,j)$$

where:

$$ST(i,k,j) = D(i,k) + D(j,k) - D(i,j).$$

The best place to insert unrouted customer P(k) is between the two adjacent customers P(i) and P(j) (both already in the current route) such that SV(i,k,j) is a maximum for all adjacent P(i) and P(j) already in the current route. Since for any customer P(k), 2D(0,k) is a constant, this is equivalent to minimizing ST(i,k,j). This calculation is made for each unrouted customer in step one of the algorithm.

The customer P(k) to enter the route is selected such that SV(1,k,m) is a maximum and customer P(k) does not cause feasibility constraints (capacity, distance, etc.) to be violated.

Based upon the work of Gaskell, Mole and Jameson extended their savings criteria by adding two parameters to the savings calculation. This modified savings formula is:

$$MSV(i,k,j) = (LB)D(0,k) + (U)D(i,j) - D(i,k) - D(j,k)$$

where LB and U are the two parameters. This formula can be rewritten as:

$$MSV(i,k,j) = (LB)D(0,k) - MST(i,k,j)$$

where

$$MST(i,k,j) = D(i,k) + D(j,k) - (U)D(i,j).$$

Up to now the discussion has only involved inserting a customer between two customers already in the current route. Two other points need to be considered. The first involves inserting a customer between the depot and a customer already routed and the second is determining which two customers will be used to initiate the current route. To address the first point, there is no difference in inserting a customer between the depot and another customer or inserting a customer between two adjacent customers. The savings formula for inserting a customer between the depot and another customer is:

$$\begin{aligned} SV(0,k,j) &= 2D(0,k) + D(0,j) - D(0,k) - D(j,k) \\ &= D(0,k) + D(0,j) - D(j,k) \\ &= S(j,k). \end{aligned}$$

The modified formula is:

$$\begin{aligned} MSV(0,k,j) &= (LB)D(0,k) + (U)D(0,j) - D(0,k) - D(j,k) \\ &= (LB-1)D(0,k) + (U)D(0,j) - D(j,k). \end{aligned}$$

To initiate a route any of several selection criteria may be used to select the first customer(s). Possibilities include selecting the unrouted customer furthest from the depot, or the unrouted customer with the largest demand. Another possibility is to select the unrouted pair of customers

that are feasible and when put into the route, yield the greatest savings using the Clark and Wright formula for savings.

The optional fourth step attempts to look for potential improvements by moving customers between routes and then trying to eliminate a route by redistributing customers from the least capacity route to all the other routes. The criterion used to move a customer is based on the value of either ST(i,k,j) or MST(i,k,j). If ST or MST can be made smaller than its current value by moving customer P(k) to another route, and customer P(k) can feasibly be added to this new route, then the transfer is made. This procedure is employed to overcome a basic problem of sequential route building. Sequential methods tend to add customers to the current route simply because a load or distance constraint has not been met, even if it may be expensive to do so.

After every customer has been checked for potential rerouting, an attempt is made to reduce the number of routes by redistributing the customers on the smallest laden route among the remaining routes. If this can be done feasibly, the first part of the refinement procedure is used again to look for improvements by rerouting customers. The procedure ends when no further reductions in routes can be made.

The sample problem of Section D is now solved using this algorithm. Initiate route 1 by selecting the customer furthest from the depot; this is customer P(2). Since there is only one customer in the current route, any customer added to the

route must be placed between customer P(2) and the depot. Maximizing SV(0,i,2) over P(i) where i is an element of {1,3,4,5} is equivalent to maximizing S(i,2) over P(i) where i is an element of {1,3,4,5}. From Table III-II, i = 1 maximzies SV(0,i,2). Customer P(1) is added to route 1 and route 1 is closed because there is not a truck with sufficient capacity to service the additional demand of any of the remaining customers.

Route 2 is initiated by selecting the unrouted customer furthest from the depot which is customer P(3). Maximizing S(i,3) over P(i) where i is an element of {4,5} results in a maximum for i = 4. Route 2 is now P(0)-P(4)-P(3)-P(0). Since in this example only customer 5 remains unrouted, it will be inserted into route 2 is it is feasible to do so. It is found that ST(i,5,j) is a minimum for i = 3 and j = 0. The sequencing of the customers in route 2 cannot be improved. Thus route 2 is P(0)-P(4)-P(3)-P(5)-P(0). No refinements are possible because it is impossible to move any customer from any route to any other route due to capacity constraints. The final solution is the same as that contained using the Clark and Wright algorithm.

H. HOLMES AND PARKER

In the original Clark and Wright algorithm, once two customers are linked together in a route, they cannot be unlinked and reexamined for improvements. To help improve results, Holmes and Parker [Ref. 14] suggest a procedure of

progressively eliminating high-savings links and looking for any improvements that may result.

The method starts by initially solving the VRP using the Clark and Wright algorithm and recording the order in which links are added. Call the resulting solution SL(1). The next step is to temporarily prohibit the first link added in SL(1), say link P(1*)-P(j*), from entering into a subsequent solution by temporarily setting S(i*,j*) = 0. The VRP is then solved again using the Clark and Wright algorithm and solution SL(2) is obtained. If SL(2) is better than SL(1), SL(2) becomes SL(1) and S(i*,j*) is set permanently to zero. Then the first link P(i1*)-P(j1*) added in the new SL(1) has S(i1*,j1*) temporarily set to zero and the VRP is solved again. If, however, SL(2) is not better than SL(1), S(i*,j*) is set equal to its original value, the next link added in SL(1) has its savings temporarily set to zero and the VRP is solved again. This procedrue continues until a specified number of successive temporary link suppressions yield no improvement or until all links in SL(1) have been temporarily suppressed with no resulting improvement.

Consider again the example problem of Section D. The Clark and Wright algorithm produces SL(1) = (P(0)-P(1)-P(2)-P(0); P(0)-P(5)-P(3)-P(4)-P(0) in which the first link added was P(1)-P(2). Temporarily set S(1,2) = 0 and resolve the problem. If this is done the resulting solution SL(2) = (P(0)-P(2)-P(3)-P(4)-P(0);{P(0)-P(1)-P(5)-P(0)} is obtained. The combined route distance of SL(1) is 54.5 while for SL(2)

it is 50.5. Thus SL(1) is set equal to SL(2) and S(1,2) is set permanently to zero. The first link added in the new SL(1) was link P(2)-P(3). Temporarily set S(2,3) to zero and solve the problem again. The resulting solution is SL(2) = {P(0)-P(1)-p(3)-P(0); P(0)-P(4)-P(2)-P(5)-P(0)} with a combined route distance of 61.5. The new solution is not better than SL(1), so S(2,3) is again set to its original value of 16. The second link added in SL(1) was link P(1)-P(5). Therefore this link is temporarily set to zero and the problem solved again. The procedure terminates when the last link added in the current SL(1), namely P(3)-P(4), is temporarily set to zero and no improvement results. In this example, the current solution is the final solution. It is also the same one found by the Tillman and Cochran algorithm.

I. SWEEP ALGORITHM OF GILLET AND MILLER

The sweep algorithm of Gillet and Miller [Ref. 10] is conceptually different than the savings approach. In order to use this method, not only is the matrix of interstop distances required, but geographical coordinates for each customer are also needed. The idea is to sequentially form routes by sweeping through an ordered list of customers. Once a route has been formed, potential improvements are sought by attempting to add customers to or delete customers from the route. When there are no more possible improvements, the next route is formed and the process is continued until all customers

have been routed or until no further customers can be routed. After the final route has been formed, a new initial ordering of customers is used to initiate the process again.

The ordering of customers is the key to the method. A customer is selected as a reference to form a base direction with the depot which is the origin. Then all customers are ordered according to the polar coordinate angle that they form with this base direction. In other words, an imaginary arm is pivoted around the depot starting with the base customer. Customers are added to the ordered list as the pivot arm passes over them.

The first step is to form routes, one at a time, by adding customers to the current route from the ordered list until no further customers can be feasibly added. Periodically, a travelling salesman algorithm is used on the current route to optimize the sequence of deliveries. If this is not done, the distance constraint may force the formation of a new route before it should.

When no further customers can be added to the current route, an improvement process (the second step) is used that looks for ways of adding or deleting customers. Assume that P(1) was the last customer added to the current route. A search is then made to find the unrouted customer P(n) that is nearest to customer P(1). P(n) is not necessarily the customer next on the ordered list. If P(n) can be feasibly added to the current route, and it is advantageous (this term is explained below) to do so, it is put into the route. If

P(n) is added to the current route, P(1) is updated (P(1) on any route is defined to be the customer with the greatest polar coordinate angle) and this step is repeated.

The term "advantageous" used in the description of the algorithm means that a changed route will result in a smaller total combined distance for the current route and the route to be formed next. This may seem strange in that the next route has not been formed yet. To overcome this problem an arbitrary number of the next unrouted customers (say 5 or 6) from the ordered list are selected. A travelling salesman algorithm is used to determine the minimum distance of a route through these unrouted customers. Call this distance D2. D1 is the minimum distance of the current route. Next assume that the change has temporarily taken place. Recalculate these two minimum distances and call them D1' and D2'. If D1+D2 > D1'+D2' then the change is advantageous, otherwise it is disregarded.

If P(n) cannot be added to the route, the best customer P(r) to be removed from the route is found. The best customer for removal is the customer that will be best (out of the customers on the current route) included in the next route. Such a customer should be close to the depot and close to the customers that will form the next route. P(r) can be found by minimizing the function D(0,j)-AN(j)(AD) over all customers P(j) on the current route, where AN(j) is the polar coordinate angle of customer j and AD is the average distance of all customers from the depot. If feasible and advantageous, P(r)

is replaced by P(n), P(1)is updated as before and the previous step is repeated. Otherwise, the final step is initiated.

The final refinement step involves investigating the possibility of replacing P(r) by two customers P(n) and P(s) where P(s) is the second nearest customer to P(1). Obviously if replacing P(r) by P(n) in the previous step was not feasible, this step will not be feasible either and can be skipped. However, if the previous step was feasible, but not advantageous, this step may be feasible and advantageous. If the replacement of P(r) by P(n) and P(s) is made, P(1) is updated and a return is made to the second step. Otherwise, the current route is complete and a return is made to the first step where the first unrouted customer in the ordered list starts the next route. If there are no more trucks to route or all customers are routed, the first iteration of the algorithm is complete.

After each iteration of the algorithm the base direction is rotated by one customer until all possible forward rotations have been considered. Once all of the forward rotations have been considered, the direction of rotation can be changed and all possible backward rotations considered. In general each forward and backward rotation will yield different results. The best routing of all those considered is selected.

As an example of this algorithm, again consider the problem of Section D. Each customer has cartesian coordinates and polar coordinate angles as given in Table III-IV. The

TABLE III-IV

CUSTOMER	X	Y	Angle in radians referenced to P(5)
P(0)	0	0	
P(1)	5	2	.566
P(2)	4	3	.79
P(3)	1	4	1.52
P(4)	-5	2	2.95
P(5)	5	-1	0

angles are in radians and are with reference to customer P(5). The algorithm might actually try other customers as the reference direction before finding that customer P(5) will produce the best solution. The algorithm starts forming route 1 by progressively adding customers from the ordered list until a feasibility condition is violated. Customers P(5) and P(1) are thus added before the capacity of the largest vehicle (1950 in this case) is exceeded. Thus at the end of step 1, route 1 is P(0)-P(5)-P(1)-P(0).

Step 2 of the algorithm now looks for the unrouted customer nearest P(1), the last customer added to route 1 and tries to add this customer to route 1. This customer is found to be P(2). However, customer P(2) cannot be added to the route without exceeding the vehicle's capacity.

Step 3 finds the customer on route 1 best suited for replacement by P(2) by minimizing D(0,i)-AD(AN(i)) over P(i) an element of route 1. This expression is minimized for

P(1). Only three unrouted customers remain, so the decision of whether to replace P(1) with P(2) is made by comparing the total route distance of 50.5 for {P(0)-P(5)-P(1)-P(0); P(0)-P(2)-P(3)-P(4)-P(0)} with 60.5 for {P(0)-P(5)-P(2)-P(0); P(0)-P(1)-P(3)-P(4)-P(0)}. Note that each route listed is optimally sequenced. This comparison leads to the decision to keep P(1) in route 1.

The fourth step investigates replacing P(1) with the first and second nearest unrouted customers to P(1). The second nearest customer is found to be P(3). Therefore a comparison is made between {P(0)-P(5)-P(1)-P(0); P(0)-P(2)-P(3)-P(4)-P(0)} and {P(0)-P(5)-P(2)-P(3)-P(0); P(0)-P(1)-P(4)-P(0)}. The latter's total route distance is 58.5 and therefore is kept in route 1. Route 1 is now established and the next route is formed by returning to step 1. The final routes are P(0)-P(5)-P(1)-P(0) and P(0)-P(2)-P(3)-P(4)-P(0).

J. TREE SEARCH

The tree search algorithm [Ref. 3: p. 313-317] is an exact method and theoretically can yield the optimal solution to the vehicle routing problem. However, with increases in the number of customers, the method quickly becomes impractical because of the computation time involved. Thus, heuristic tree search methods have been developed based upon the exact method. One such heuristic is explained in this section. However, to understand the heuristic, it is helpful to understand the concept of the exact method first.

The exact tree search algorithm is a depth-first search method whereby every feasible routing of vehicles is or can be explored by going through a methodical process. The search starts by setting the objective value of the incumbent set of routes to infinity or to the value of the best known solution. An incumbent set of routes is defined as the best (in terms of the objective) set of routes found at any stage during a search. If at some point in the search a better set of routes is found, this new set of routes becomes the incumbent set. As will be explained later, the search can be limited somewhat by using bounding and feasibility arguments.

Initially none of the customers are routed. A customer is selected and a route (also called a node) containing this customer is formed that can be serviced by at least one truck in the fleet. Usually there are many such routes that can be formed, all of which will be investigated eventually. Once this route is formed there are two distinct sets of customers, those that have been routed and those that have not been routed. From among the unrouted customers another customer is selected. A route containing this customer that can be feasibly serviced by at least one truck in the fleet is then formed. This process of forming routes from unrouted customers is continued until either all customers have been routed or until it is not feasible to route any further customers. At this point one branch of a tree has been formed where each node of the branch represents a route serviced by

a single vehicle. In Figure 7 one such branch is depicted where the candidate solution CS = {R(1,k)1)),R(2,k(2)),..., R(n,k(n))}. In the notation R(i,k(i)), i refers to the ith stage and k(i) refers to the k(i)th route formed at the ith stage.

Following back up the branch, none of the customers in any route R(i,k(i)) can be found in any other route R(j,k(j)). Since the criterion for forming a route is that at least one vehicle from the fleet be able to service the entire route, it is possible that there is not a feasible assignment of vehicles to routes. Thus, at each stage after a new route has been formed a check must be made to determine if there is a feasible allocation of trucks to routes. If no such allocation exists, the most recently added route is removed and not considered again.

When a search ends at stage n along a branch because there are no more customers to route, a candidate solution has been found. The candidate set of routes is compared with the incumbent set of routes. The set of routes with the best objective value is kept and the other set deleted. Next, a backtracking process begins. The route R(n,k(n)) is replaced with route R(n,k(n)+1) (if R(n,k(n)+1) exists) and the search is continued. If at stage n, R(n,k(n)+1) does not exist or bounding or feasibility arguments prove the optimal solution does not lie along the current branch, a return is made to stage n-1 and R((n-1),k(n-1)) is replaced with route R((n-1),k(n-1)+1) and the search is continued.

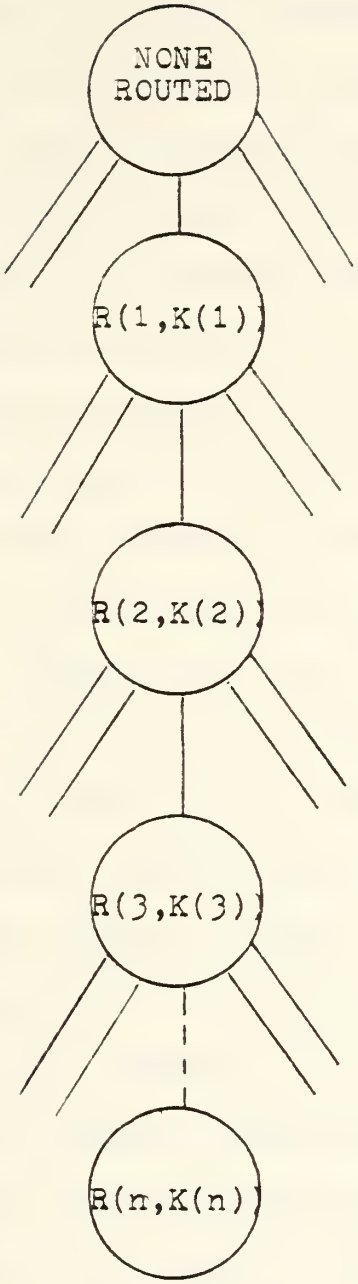


FIGURE 7

The above search procedure can be narrowed as mentioned earlier by using feasibility and bounding arguments. The last node added along the current branch can be removed from further consideration if one of the following conditions occurs: 1) trucks cannot be feasily assigned to routes; 2) the cost of all routes currently formed plus a lower bound on the cost of routing the unrouted customers is greater than the best solution found so far; 3) the remaining vehicles have insufficient capacity to service the unrouted customers; 4) a lower bound along the current branch is greater than an upper bound along another branch.

In Table III-V the interstop distances for a three-customer problem are given. Suppose that two trucks each with a capacity of 10 are available to service the demand of each customer listed in column Q of Table III-V. The solution of this problem using the exact tree search method is depicted in Figure 8. The circles represent nodes or routes and the numbers in the circles are the customers on the respective route. The nodes of stage 1 collectively exhaust all feasible single-vehicle routes. The nodes of stage 2 represent feasible single-vehicle routes given that the corresponding route at stage 1 has already been created. The numbers under the nodes of stage 2 are the total travel distance of the respective branch and NF indicates that the given branch is not feasible. The branches with the best objective value corresponds to the branches consisting of the routes P(0)-P(1)-P(0) and P(0)-P(2)-P(3)-P(0).

TABLE III-V

Q	P(0)		
5	5	P(1)	
5	6	5	P(2)
5	7	7	4 P(3)

(a)

Trucks

Capacity	10
Available	2

(b)

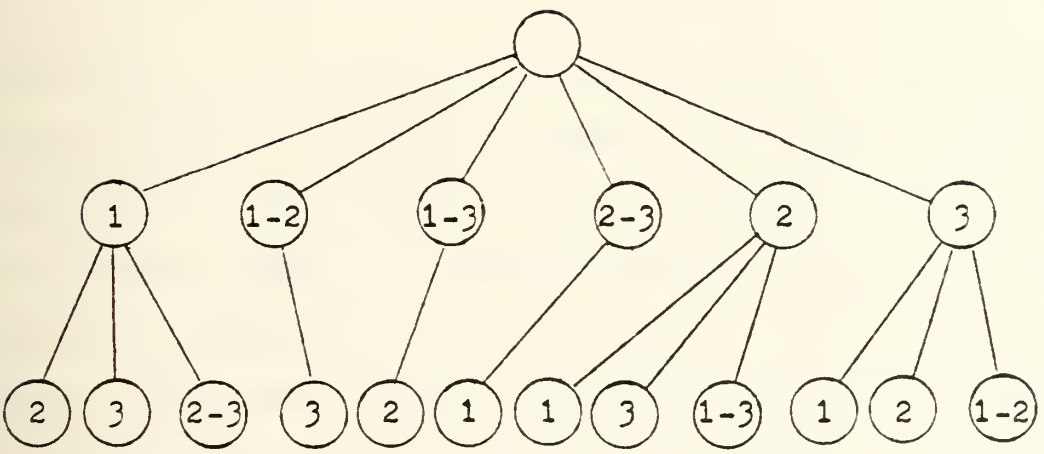


FIGURE 8

The concept of the heuristic tree search algorithm is to limit the search process to branches that show promise of yielding routes with low objective values. Thus, all branches are not considered and the resulting solution may not be optimal.

In the exact algorithm, at any stage i, all branches leading from the current node at stage i are investigated. The heuristic algorithm, however, only generates a small portion of these branches. A measure that can be used in generating these branches is a varying linear combination of the savings criteria of Clark and Wright and an extra mileage criterion. The formula is:

$$G(i,j) = S(i,j) - (U)EM(0,i,j)$$

where EM(0,i,j) is the minimum extra mileage associated with either inserting unrouted customer P(i) between routed customer P(j) and the depot or P(j) between routed customer P(i) and the depot. The extra mileage is determined from $EM(0,i,j) = \min[D(0,i) + D(i,j) - D(0,j);\ D(0,j) + D(i,j) - D(0,i)]$. Several (say five or ten) candidate routes can be formed from unrouted customers by varying the U parameter. Each route is formed using the sequential route-forming logic of Mole and Jameson. However, only one route is formed for each U value.

Once several candidate routes have been created, it is necessary to evaluate them for inclusion in the final set of routes. This evaluation cannot be exact because how good a

route is depends on how good future routes formed by the algorithm will be. Therefore a quick estimate of the future routing distance for all potential candidates needs to be found. This estimate can be obtained by solving the minimum spanning tree problem (see Appendix B). The process works by finding the minimum spanning tree rooted at the depot for customers that would be left unrouted if the current candidate were to be put into the final set of routes. The minimum spanning tree is used because it finds a quick lower bound and therefore an approximation to the optimum routing of the remaining customers. The total distance of the minimum spanning tree can be found quickly and added to the distance of the current candidate. Thus the following is calculated for each candidate:

$$BR(i) = DR(i) - TF(i)$$

where DR(i) is the total distance of candidate route R(i) and TF(i) is the minimum spanning tree distance of the unrouted customers F(i) that would be left if R(i) were to be routed. The candidate route with the smallest value of BR(i) is routed and the whole process is started again at the next node in the tree. The algorithm stops when either no further customers can be added feasibly or when all customers have been routed.

The problem of Section D will now be solved using this heuristic. In Table III-VI, G(i,j) has been calculated for

TABLE IV-VI

LINK	S	EM	G U = 1	G U = 4
1-2	19	1	(1) 18	(2) 15
2-3	16	0	(2) 16	(1) 16
1-3	11	5	(4) 6	(4) -9
1-5	10.5	.5	(3) 10	(3) 8.5
3-4	6	6	(5) 0	(6) -18
2-5	5.5	5.5	(6) 0	(5) -16.5
2-4	5	7	(7) -2	(7) -23
3-5	3.5	7.5	(8) -4	(8) -26.5
1-4	3	9	(9) -6	(9) -33
4-5	.5	10.5	(10) -10	(10) -41.5

every link P(i)-P(j) and for values of U = 1 and 4. The numbers in parenthesis in columns four and five represent the rank order (from highest to lowest) of the associated G value. For a parameter value of U = 1, the highest value link is link P(1)-P(2). When link P(1)-P(2) is added, the candidate route P(0)-P(1)-P(2)-P(0) is formed (call this CR1). No further customers can be added to CR1 because of the vehicle capacity constraint. For a parameter value of

U = 4, the highest value link is link P(2)-P(3). After this link is added, the next feasible link that will add customers to this route is link P(3)-P(4). Thus CR2 = P(0)-P(2)-P(3)-P(4)-P(0) is formed. The distance of CR1 is D1 = 25 and the distance of CR2 is D2 = 30.

Next the minimum spanning tree distance for the unrouted customers is found for each of these cases. The minimum spanning tree distance associated with CR1 is MS1 = 19.5 and with CR2 is MS2 = 10.5. Comparing D1 + MS1 = 44.5 with D2 + MS2 = 40.5, CR2 is found to be the better route. When CR2 is kept as the first route the second route is found to be P(0)-P(1)-P(5)-P(0) and the optimal solution results.

K. A TWO-PHASE ALGORITHM

The two-phase algorithm [Ref. 3: p. 313-337] consists of two phases that are used over and over again until all customers have been routed or until no further customers can be routed. The purpose of the first phase is to determine key customers, each of which will start the formation of a route in the second phase. These key customers are found by going through an initial-route building procedure. The key customers are then the customers that initiate each of these routes. The second phase then sequentially forms routes based upon the key customers until all customers are routed, or until no further customers can be placed in routes. At the end of the second phase, if unrouted customers still remain, the algorithm returns to phase 1 with the unrouted customers.

In phase 1 routes are formed sequentially by first selecting an unrouted customer to start the formation of a route. Call the current route being formed R(i). Any of several criteria may be used to select the initial customer. A possibility is to pick the unrouted customer furthest from the depot. Assume now that the customer to start the next route has been chosen and is P(a). Now for each unrouted customer P(j), calulate the following:

$$K(j) = D(0,j) + (U)D(a,j)$$

where U is an arbitrary parameter greater than 1.

Next pick the customer P(j*) that has the smallest K value and put this customer into the current route. Optimize the sequence of the customers in the current route by solving the travelling salesman problem. Continue placing customers in the current route by selecting the feasible customer with the smallest K value. Solve the travelling salesman problem each time a customer is added to the route. This step is continued until there are no more customers that can be feasibly added to the current route.

Next close the current route and select an unrouted customer to form a new route as before. This process is continued until all customers are routed or until all remaining customers cannot be routed. At this point phase 1 of the method ends and phase 2 begins. At the end of phase 1, there were a number of routes (say h) that were formed. Each of the customers (called key customers) that were selected to

initially form each of the h routes in phase 1 is now selected to form h single-customer routes. Thus at the beginning of phase 2 there are h routes of the form depot-customer-depot.

The first step in phase 2 is to associate each of the now unrouted customers with one of the key customers found in phase 1. In making this association, unrouted customer P(i) should be matched with a routed customer P(k) that is far from the depot and close to P(i). This is done by determining the following value for each unrouted customer P(i) and routed customer P(k):

$$V(i,k) = D(0,i) + (U)D(i,k) - D(0,k)$$

where U is greater than or equal to 1. Call V an association coefficient. Each customer P(i) is then best associated with that route for which the association coefficient is smallest. At the same time that the best association for P(i) is being found, the second best association is also found. The second best association is with routed customer P(k') such that V(i,k') is second smallest.

At this point there are h single-customer routes and all unrouted customers are associated with one of these routes. However, it is not certain now whether or not it is feasible to put all the unrouted customers into their associated routes. Therefore each route must have an ordered list of its associated customers that can be brought in one at a time until no further customers can be added. This ordering is done by picking that customer associated with a given route that has

the highest difference in "association" between its second best associated route and its best associated route.

The procedure of ordering the associated customers involves first picking one of the routes (say route R(k)). The following value is then calculated for each customer P(i) associated with R(k):

$$DL(i) = V(h',i) - V(h,i).$$

Here, P(h') is the key customer to which P(i) has its second best association. Then the first associated customer to enter the current route is that customer for which DL(i) is a maximum.

Each time a customer is brought into the route, the route is resequenced with a travelling salesman algorithm. Customers are added to the current route using the max DL criteria until none of the remaining customers are feasible. This step is completed for each of the routes and the associated customers. At this point phase 2 of the algorithm is complete. If unrouted customers remain at the end of phase 2, the algorithm returns to phase 1 with the unrouted customers. Otherwise it terminates.

This algorithm will now be demonstrated on the example problem of Section D. Initialize route 1 of phase 1 by selecting the customer furthest from the depot. This is customer P(2) from Table III-I. The value of K(j) for each unrouted customer P(j) is calculated. Using a U value of 1, K(1) = 13, K(3) = 12, K(4) = 19 and K(5) = 17.5 Customer

P(3) enters route 1 first because K(3) is smallest. Customer P(1) cannot enter next because of the capacity constraint. Customer P(5) enters route 1 next and is the last customer to enter, again because of vehicle capacity. Route 1 is P(0)-P(2)-P(3)-P(5)-P(0) after customers are optimally sequenced. The above procedure yields P(0)-P(1)-P(4)-P(0) for the second route where P(1) initializes route 2.

The two routes P(0)-P(2)-P(0) and P(0)-P(1)-P(0) are formed to initiate phase 2. Values of the association coefficients are found as follows:

	Association Coefficients			
	Route 1	Route 2	Difference	Route Association
P(3)	0	5	5	1
P(4)	7	9	2	1
P(5)	5.5	.5	5	2

Based on the above, P(3) enters route 1 first because the difference between its second smallest and smallest association coefficients is largest. Customer P(4) enters second. Customer P(5) enters route 2. Customers are optimally sequenced and routes P(0)-P(1)-P(5)-P(0) and P(0)-P(2)-P(3)-P(4)-P(0) are thus formed.

L. GIANT TOUR

The method of the giant tour algorithm [Ref. 17: p. 14-17] is to form a single route (the giant tour) and then break up

this giant tour into feasible subtours, each of which can be serviced by a truck. The giant tour can be formed by using a travelling salesman algorithm (see Appendix A).

The problem of breaking up the giant tour can be solved by transforming it into a shortest path problem (see Appendix C for an explanation of the shortest path problem). The TSP tour can be written as P(0)-P(K(1))-P(K(2))-...-P(K(n))-P(0) where K(i) is the index for the ith customer in the tour. For any two customers P(K(i)) and P(K(j)), i less than j, the transformed distance between them, D'(K(i),K(j)) is set equal to the distance of the subtour P(0)-P(K(i+1))-P(K(i+2))-...-P(K(j))-P(0) if this subtour is feasible. If it is not feasible, D'(K(i),K(j)) is set equal to infinity. The process of creating the transformed matrix is a device whereby every feasible way of breaking up the giant tour is enumerated.

The result of the above transformation is a distance matrix where the entry of row P(K(i)) and column P(K(j)) represents the distance of route P(0)-P(K(i+1))-P(K(i+2))-...-P(K(j))-P(0). The shortest path problem is then solved to find the shortest path from the depot to customer P(K(n)). The arcs of the shortest path can be quickly transformed into a solution to the vehicle routing problem. Arc (P(K(i)),P(K(j))) of the solution to the shortest path problem corresponds to a route P(0)-P(K(i+1))-P(K(i+2))-...-P(K(j))-P(0).

Figure 9 illustrates this concept graphically. The transformed distance D'(k(0),k(2)) is equal to the distance of the route P(0)-P(k(1))-P(k(2))-P(k(0)). Likewise, the

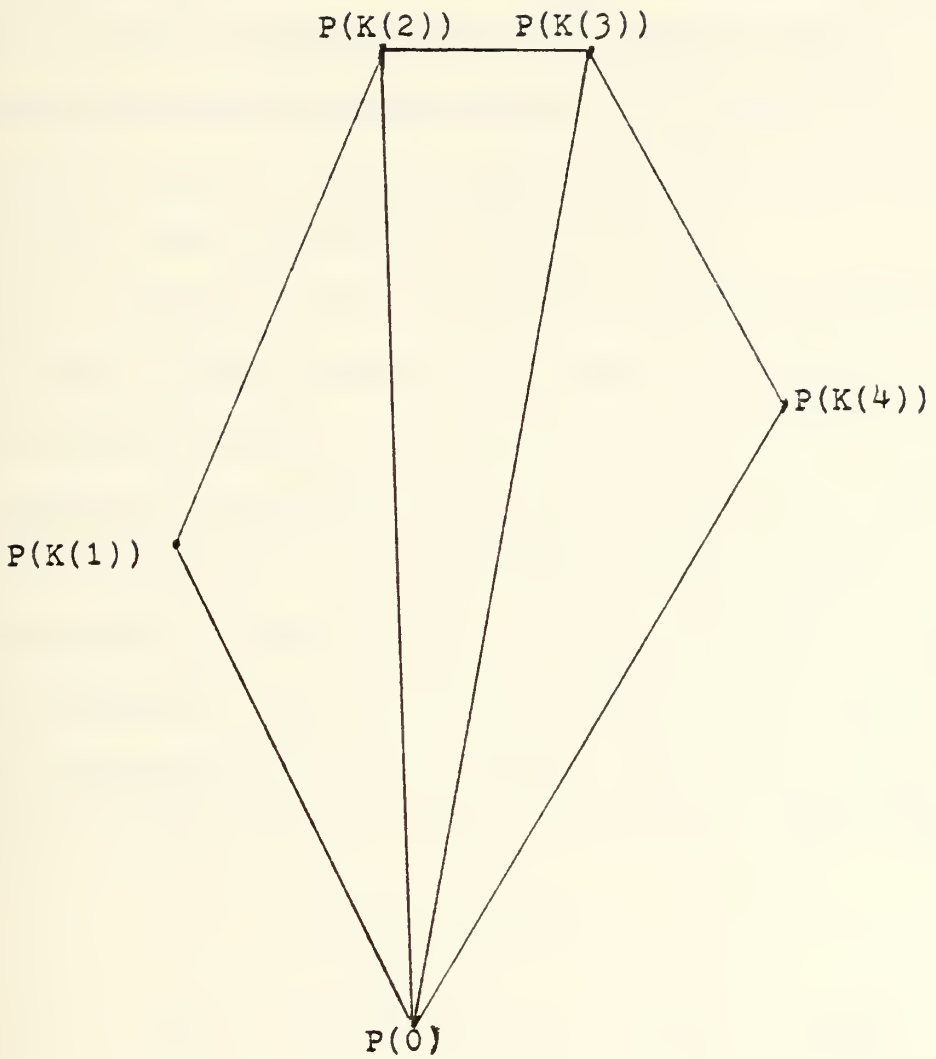


FIGURE 9

transformed distance D'(k(1),k(3)) is equal to the distance of the route P(0)-P(k(2))-P(k(3))-P(0). If in either of these cases the route is infeasible, the respective transformed distance is set equal to infinity. If the solution to the problem of finding the shortest path between P(0) and P(k(4)) using the transformed distance matrix is found to be P(0)-P(k(2))-P(k(4)), the corresponding routes are P(0)-P(k(1))-P(k(2))-P(0) and P(0)-P(k(3))-P(k(4))-P(0).

If all customers from the example problem of Section D are put into a single route (giant tour), the optimal sequence of the customers is P(0)-P(4)-P(3)-P(2)-P(1)-P(5)-P(0). This sequence of customers is found by solving the TSP. Using this giant tour, a transformed distance matrix is found as shown in Table III-VII. Entry (P(0),P(2)) corresponds to feasible route P(0)-P(4)-P(3)-P(2)-P(0) which has a route distance equal to 30. Entry (P(4),P(2)) corresponds to feasible route P(0)-P(3)-P(2)-P(0) with a route distance

TABLE III-VII

		P(4) P(K(1))	P(3) P(K(2))	P(2) P(K(3))	P(1) P(K(4))	P(5) P(K(5))
P(0)	P(K(0))	12	22	30	∞	∞
P(4)	P(K(1))	∞	16	24	∞	∞
P(3)	P(K(2))	∞	∞	24	25	∞
P(2)	P(K(3))	∞	∞	∞	20	20.5
P(1)	P(K(4))	∞	∞	∞	∞	11

of 24. Entries corresponding to infeasible routes are infinity.

Once the transformed matrix has been created, the shortest path between the depot and the last customer of the giant tour (P(5)) is found using this matrix. The shortest path is found to be P(0)-P(2)-P(5) with a path distance of 50.5. This shortest path corresponds to routes P(0)-P(4)-P(3)-P(2)-P(0) and P(0)-P(1)-P(5)-P(0).

M. R-OPTIMAL ALGORITHM

The r-optimal algorithm [Ref. 7] is based upon the r-optimal travelling salesman heuristic of Lin [Ref. 8: p. 132-135] (see Appendix A). A feasible solution S to the VRP is said to be r-optimal if no improvement (decreased time or ditance) can be made by replacing r links of S by r other links such that feasibility is maintained. This algorithm is thus an improvement heuristic whereby an initial feasible solution is improved by making feasible exchanges of r links with r other links such that the new solution is feasible and has an improved objective value.

Before the algorithm proceeds, the matrix of interstop distances (times or costs) needs to be transformed. Suppose that an initial solution with w routes is generated. Then replace the original depot in the matrix with w artificial depots each of which is the same distance as the original depot from each customer. Furthermore, to prevent one artificial depot from being linked to another artificial depot,

make all interdepot distances infinity. This transformation has been made on the matrix of Table III-I and is given in Table III-VIII under the assumption the two routes P(0)-P(1)-P(2)-P(0) and P(0)-P(5)-P(3)-P(4)-P(0) have been generated as the initial solution. The w routes of the initial solution can now be transformed into a single tour in which each route of the initial solution starts at one artificial depot and ends at another artifial depot. For example, the initial set of routes selected for the current example can be written as P(01)-P(1)-P(2)-P(02)-P(5)-P(3)-P(4)-P(01) where P(01) and P(02) are the artificial depots.

Given a single tour in which there are n customers, there are C = n!/(r!(n-r)!) ways of choosing r links. Thus, to prove that a given tour is r-optimal, C combinations of links must be checked for possible replacements. The step-by-step procedure follows.

TABLE III-VIII

	P(01)	P(02)	P(1)	P(2)	P(3)	P(4)	P(5)
P(01)	∞	∞	10	12	8	6	5.5
P(02)	∞	∞	10	12	8	6	5.5
P(1)	10	10	∞	3	7	13	5
:(2)	12	12	3	∞	4	13	12
P(3)	8	8	7	4	∞	8	10
P(4)	6	6	13	13	8	∞	11
P(5)	5.5	5.5	5	12	10	11	∞

Step 1. Generate a feasible solution to the VRP. This can be done by using any of the other heuristics described in this chapter, by hand or randomly. If a random solution is generated, this method can be used with several different initial feasible solutions, and the resulting r-optimal solution with the best objective value selected. Transform the interstop matrix as described above and identify the initial single tour. Set COUNT = 0.

Step 2. Select r links for temporary removal. Add one to COUNT.

Step 3. Insert r new links such that feasibility is maintained.

Step 4. If the new set of links results in an improvement, permanently replace the old set with the new set, set COUNT = 0 and go to Step 2. Otherwise continue on to Step 5.

Step 5. If all feasible ways of replacing the original r links have not been considered, go to Step 3. Otherwise continue to Step 6.

Step 6. If COUNT = C, the r-optimal solution has been found and the algorithm terminates. Otherwise go to Step 2.

An r-optimal solution where r is equal to the number of customers is the optimal solution. Unfortunately, this method is prohibitive in terms of computation time with problems of any size for values of r greater than 3.

An interesting property of the transformed interstop matrix needs to be mentioned. If all interdepot distances

are set to infinity, the final r-optimal solution will have the same number of routes as the initial solution. If however, the interdepot distances are set to zero, the algorithm will reduce the number of routes, if this is consistent with reduced total mileage. This property of the transformed distance matrix allows one the flexibility to minimize the distance (time or cost) subject to using a specified number of trucks or to minimize total distance without such a restriction. It should be pointed out that the minimum-mileage set of routes may not have the minimum number of routes.

An example problem using this algorithm is worked in Section N.

N. TIME WINDOWS [Ref. 22]

Consider a problem in which a set of customers S (where S is a subset of all customers) have an early time and a late time (both of which fall within delivery period T) during which delivery must be made. Call the time between the early time and the late time a time window. Given an initial feasible solution to this problem, the r-optimal algorithm can be used to improve upon the initial value of the objective while maintaining feasibility. An initial feasible solution to this problem can be found by initially routing all customers while disregarding the time window requirements. If all time window restrictions are met, an initial solution is found. If the time windows for some customers are not met, start removing those customers (one

at a time) for which time windows are violated until all routes are feasible. Then, for each customer thus removed, route one truck from the depot to that customer and back to the depot, such that all time window conditions are met. These single-vehicle routes may require that a truck leave the depot after time zero in order to arrive within a given time window. With this initial feasible solution, the r-optimal algorithm can now be employed.

As an example, consider the problem for which interstop times and customer demands are given in Table III-IX. In this problem, customer P(2) must receive delivery between time 0 and time 10. All other customers have no time windows. The delivery period starts at 0 and ends at 30. Two trucks, each with a capacity of 20 are available. Assume an initial solution has been found using the procedure described above and that it is P(0)-P(1)-P(3)-P(0), P(0)-P(2)-P(0) and P(0)-P(5)-P(4)-P(0). Table III-IX includes the three artificial depots

TABLE III-IX

	Q	P(01)	P(02)	P(03)	P(1)	P(2)	P(3)	P(4)	P(5)
P(01)	0	0	0	0	5	7	7	8	4
P(02)	0	0	0	0	5	7	7	8	4
P(03)	0	0	0	0	5	7	7	8	4
P(1)	10	5	5	5	∞	10	6	13	8
P(2)	5	7	7	7	10	∞	5	4	3
P(3)	5	7	7	7	6	5	∞	10	7
P(4)	10	8	8	8	13	4	10	∞	3
P(5)	5	4	4	4	8	3	7	3	∞

corresponding to these three routes. The initial solution can be written in terms of the artificial depots as P(01)-P(1)-P(3)-P(02)-P(2)-P(03)-P(5)-P(4)-P(01). Notice that the initial solution has more routes than vehicles available. This difficulty is overcome during the improvement process.

If 2-optimal procedures are employed, a search must be made for two links that can be removed and replaced by two other links that result in less total travel time but maintain feasibility. If links P(01)-P(1) and P(02)-P(2) are removed and replaced with links P(01)-P(02) and P(1)-P(2) an improved feasible route is found. The new sequence of customers is P(01)-P(02)-P(3)-P(1)-P(2)-P(03)-P(5)-P(4)-P(01) which corresponds to routes P(0)-P(2)-P(1)-P(3)-P(0) and P(0)-P(5)-P(4)-P(0). Another search finds that replacing links P(3)-P(02) and P(2)-P(1) with links P(02)-P(1) and P(3)-P(2) will result in the improved sequence P(01)-P(02)-P(1)-P(3)-P(2)-P(03) -P(5)-P(4)-P(0). This sequence of customers corresponds to routes P(0)-P(2)-P(3)-P(1)-P(0) and P(0)-P(5)-P(4)-P(0) which is the optimal solution.

It should be noted that when checking a route for feasibility, that there are two directions in which a route can be traversed. Therefore, the time of arrival at each customer must be checked for both directions of travel before concluding that a time window has been violated. This point can be demonstrated with route P(0)-P(2)-P(3)-P(1)-P(0), which is feasible. However, the route P(0)-P(1)-P(3)-P(2)-P(0) is not.

O. A MANUAL METHOD

According to Doll [Ref. 5], most of the improvement that results from computerized vehicle scheduling results from the scrutiny that the whole delivery process comes under when the system is implemented. He argues that any well organized manual method of developing schedules will result in solutions as good as or better than those produced by heuristics. His method is based on the fact that good routes do not overlap themselves or other routes and tend to be teardrop shaped.

The procedure for this method follows:

1. Estimate the number of schedules S required. Schedules are the number of delivery trucks leaving the depot throughout the day. Thus, a truck which leaves in the morning, returns to the depot, and leaves again in the afternoon counts as two separate schedules. The number of schedules is a function of customer demand and truck capacity.

2. Estimate the number of required vehicles V to service all customers. This number is a function of the vehicle daily distance or time constraint.

3. Next determine any geographical features (such as rivers) that will force a boundary between routes.

4. Starting with any boundary created above (or any artificial boundary) start building a feasible route as much like a tear drop as possible. Avoid having the route cross itself unless this is necessary because of one-way streets.

5. Directly adjacent to the route just formed start constructing another route.

The routes formed in steps four and five above must be feasible. The total number of routes should equal S and the total number of vehicles used should equal V.

Formulas for estimating S and V follow:

$$S = N/C$$

$$V = N(T1)/L$$

$$T1 = (A)(T2)/C + B\sqrt{(E)(T2)}/N + F/C + G$$

where:

S = number of schedules

N = number of customers

C = average number of customers per schedule

V = number of vehicles

T1 = average travel time per day per customer

L = length of the delivery period

A = 1.8, B = 1.1 (found by regression)

T2 = average depot to customer time

E = length of one side of the smallest square containing the delivery area

F = fixed time associated with a schedule (driver's lunch, coffee breaks, etc.)

G = average service time per customer.

Consider the problem for which the depot to customer times and customer demands are given in Table III-X. The

TABLE III-X

CUSTOMER	TIME FROM DEPOT (HRS)	DEMAND
P(1)	.25	2
P(2)	.5	7
P(3)	.2	10
P(4)	.1	6
P(5)	.75	13
P(6)	.3	4
P(7)	.4	1
P(8)	.1	1
P(9)	.2	2
P(10)	.5	4
P(11)	.6	6
P(12)	.25	1

trucks available have a capacity of 20. The average demand per customer is 4.75. Thus the average number of customers per schedule is 20/4.75 which is approximately 4. If the fixed time per route is .5 hours, the average time per customer is .25 hours, the length of the delivery period is 8 hours, and the length of one side of the smallest square containing the delivery area is 15 miles, then the following can be calculated using the above formulas:

$$S = N/c = 3$$

$$T1 = .74$$

$$V = (12)(.74)/8 = 1.1 \text{ rounded up } = 2.$$

Thus, three schedules and two trucks are needed to service all customers. Routes are formed using a map and the procedure discussed in steps 3 through 5. Notice that even though three schedules are needed, only two trucks are required. Therefore, one truck will have to return to the depot to be reloaded before making further deliveries.

This concludes the survey of vehicle routing algorithms. The next chapter discusses implementation issues such as computational efficiency and quality of solution for each of these algorithms. Additionally, methods for obtaining data for the interstop matrix are examined.

IV. A COMPARISON OF ALGORITHMS

A. COMPUTATIONAL EFFICIENCY

A good algorithm produces solutions with small objective values across a broad spectrum of problems in a small amount of computation time. In this section each of the algorithms presented in Chapter III are evaluated with respect to these three criteria. Computation results from several sources are summarized in Table IV-I. The source of the test problems is reference 7.

The nearest neighbor algorithm is the fastest algorithm but produces solutions with high objective values and a computation time only slightly better than that of Clark and Wright [Ref. 11]. The algorithm of Cochran and Tillman, though it produces solutions with lower objective values than the Clark and Wright algorithm, requires many more comparisons before selecting a link. To illustrate this point, consider a problem with 50 customers. For such a problem there are $50!/((2!)(48!)) = 1225$ possible links to insert as the first link. The Clark and Wright algorithm picks the link with the highest savings. If the savings are ordered, the link selectio process of that algorithm picks the first feasible link from the ordered link list. On the other hand, the Tillman and Cochran algorithm (assuming all links are feasible) requires all 1225 comparisons before making the first link selection. Thus for problems of any size, the computation

TABLE IV-I

ROUTE DISTANCE

	No. of Nodes		
	50	75	100
Clark & Wright (Ref. 2,6)	585	900	886
3-optimal (Ref. 6)	556	876	863
Sweep (Ref. 2)	532	874	851
Giant Tour (Ref. 20)	636	958	969
Holmes and Parker (Ref. 12)	573	886	876
Mole and Jameson (Ref. 2)	575	871	851
Heuristic Tree Search (Ref. 2)	534	871	851
Two-Phase (Ref. 2)	547	883	851
NO. OF ROUTES			
Clark & Wright	6	10	8
3-optimal	5	10	8
Sweep	5	11	8
Giant Tour	6	11	8
Holmes and Parker	5	10	8
Mole and Jameson	5	10	8
Heuristic Tree Search	5	11	8
Two-Phase	5	11	8
COMPUTATION TIME (SEC)			
Clark & Wright	.8 (36)*	1.7 (78)*	2.4 (150)*
3-optimal	120	240	600
Sweep	12.2	24.3	65.1
Giant Tour	4.8**	10.8**	28.2**
Holmes and Parker	58***	162***	124***
Mole and Jameson	5.0	11	36
Heuristic Tree Search	7.1	15.6	38.2
Two-Phase	2.5	4.2	9.7

Notes:

The Clark & Wright, Sweep, Mole and Jameson, Tree search and Two-Phase algorithms were refined with the 2-optimal algorithm,

All times are on the CDC-6600 unless indicated otherwise,

*IBM 7090

**IBM 370/168

***Univac 1108

time becomes large in comparison with the Clark and Wright algorithm.

The method of Gaskell produces solutions that are of about the same quality as Clark and Wright but with slightly longer computation times. In his original paper, Gaskell conjectured that each particular distribution of customers would have a TH parameter value that would always produce good solutions for every problem based on those customers. In practice this has not been found to be the case [Ref. 26: p. 365].

The sweep algorithm produces fast solutions for problems that have a small number of customers in each route. It spends most of its time sequencing customers within routes. Its computation time increases linearly with the number of routes (i.e., is proportional to the number of routes), but increases quadratically with the average number of customers in a route (i.e., is proportional to the square of the average number of customers per route) [Ref. 10: p. 346]. Furthermore, the algorithm groups customers based upon a euclidian distance metric. Therefore, if the delivery area has geographical barriers (e.g., rivers, harbors, bays), the grouping of customers can be adversely affected [Ref. 3: p. 337]. For these reasons this algorithm is not universally applicable.

Beltrami and Bodin [Ref. 1: p. 68] have found that the giant tour algorithm produces good solutions in cases where there are many customers on a route and in which the side

conditions are "loose". However, Russel [Ref. 22: p. 522] has found that for all problems he has tested, including those with loose side conditions, this algorithm produces solutions with rather high objective values.

The r-optimal procedure is only practical (in terms of computation time) for values of r = 2 or 3. Christofides and Eilon [Ref. 7] have found that a procedure in which the following steps are followed works well:

1. Generate an initial random tour.
2. Use two-optimal procedures.
3. Use three-optimal procedures on the results of step 2.
4. Repeat (three to ten times) and select the best solution.

Christofides, Mingozzi and Toth [Ref. 3: p. 333-337] have tested the algorithms of Clark and Wright, Mole and Jameson, the sweep algorithm, the tree search algorithm and the two-phase algorithm on fourteen different test problems and augmented each with the 2-optimal algorithm. In problems where customer locations are random and uniformly distributed, the sweep algorithm produces solutions with the lowest objective values, but these solutions are only slightly better than the tree search and two-phase algorithms. However, in real problems, customers are not uniformly randomly distributed, but tend to be grouped together. In cases where customers are grouped together the sweep algorithm does not perform as well as either the two-phase or tree search algorithms. Moreover, the two-phase and the tree search algorithms are the most stable in that their relative performances are not

data-dependent. In terms of computation time, the Clark and Wright algorithm and the two-phase algorithm are the fastest, but the latter consistently produces solutions with better objective values.

B. DATA REQUIREMENTS

In order to implement the automated routing of vehicles, regardless of the algorithm used, the travel time between each pair of customers must be found. Times can be obtained using any of three methods. The most sophisticated method requires that a computer map of the delivery area be stored [Ref. 19: p. 250]. Each customer's location is then specified on this map and the shortest route problem is solved between each pair of customers. This method is valuable when customer locations change radically from one delivery period to the next. Also, very accurate intercustomer times can be obtained. However, the detailed work involved in producing accurate computer maps makes this method impractical for small problems.

Another method that can be used is to collect accurate time data between each pair of customers in the delivery area [Ref. 16: p. 9]. This method requires that all potential customers must be known and that their locations be fixed. Collecting this data is a huge task for problems of only moderate size. For example, a problem with only 50 customers requires that $51!/((2!)(49!)) = 1275$ times be collected for a symmetric problem. Traffic congestion as a function of time

of day and one-way streets can increase the numger of inter-customer times needed significantly.

A third technique is to approximate inter-customer times by using euclidian distances. This method requires that the cartesian coordinates of each customer be given and then each inter-customer time is assumed proportional to the associated euclidian distance. To improve upon the accuracy of this technique, barriers (such as rivers) and congested areas can be designated [Ref. 16: p. 9-11]. Each barrier B(i) is recorded with j designated crossing points, CP(i,j). The distance between two customer P(a) and P(b) on opposite sides of B(i) is then the minimum over j of D(a,CP(i,j)) + D(b,CP(i,j)).

Congested areas are areas where traffic causes vehicles to travel more slowly than is normal. A link that passes through a congested area is increased in length in proportion to the percentage of the link that passes through the congested area.

The chief advantage of this method is that the data collection effort is minimal. An accurate street map can be used along with an arbitrary grid system to locate customers. The primary disadvantage of this method is that accuracy must be sacrificed.

There are techniques that can be used to decrease the number of inter-customer times that need to be found. The first method involves creating zones into which several customers are placed [Ref. 16]. A zone is an area within which travel time can be considered negligible. The center

of each zone is used for the purpose of calculating inter-customer times. A problem involving 50 customers has 1275 potential links if all pairings are considered. If these 50 customers are grouped into 25 zones of two customers each, the total number of potential links is reduced to 325--a significant reduction in the number of links.

A second method for decreasing the number of customer links has been suggested by Golden, Magnanti and Nguyen [Ref. 11: p. 126]. Many of the possible links between customers would never be found in a solution because the travel time involved is too great. To eliminate these unlikely candidates, an arbitrary grid system can be placed over the delivery area such that each customer is contained in a rectangle with width W and height H. The set of allowable customer links then consists of the links between the depot and each customer and links between customers in the same or adjacent rectangles. The smaller the values of H and W are, the fewer the number of candidate links.

In addition to inter-customer times, there are other data required to implement automated vehicle routing. Specifically, the time to service each customer (exclusive of travel time) and the time to load each truck are needed. Customer service time may depend on how well a customer is equipped to unload trucks. Both load and unload time depend on the type of truck being used. For example, a stake truck or a flat bed can be unloaded from the side or rear, whereas a van can only be unloaded from the rear. If equipment, such as a fork lift

is being used to unload a van, a ramp must be available so that the fork lift can drive into the truck. Thus, loading and unloading of a van may take longer than a stake truck or flat bed.

Other information that must be available is the early start time and late finish time for the delivery fleet. Additionally, any customers with time window requirements must be specified along with the allowable early and late delivery times.

V. CONCLUSIONS AND RECOMMENDATIONS

The Naval Supply Centers in Oakland and San Diego operate local delivery operations that differ in two important respects. At Oakland, the entire local delivery operation is centered in one warehouse, whereas in San Deigo, the equivalent operation is located at two separate facilities about six miles apart. Additionally, in San Diego, fresh fruit and vegetables are stored in cold storage at a commercial firm. Also, the volume of material delivered in San Diego in much larger.

The local delivery problem model is applicable to both San Diego and Oakland. Although delivery is not made from one central depot in San Diego, the material stored at each of its separate facilities is different and no decisions need to be made as to which depot is going to make delivery to which customer. Therefore, there are three separate local delivery problems.

Automated vehicle routing must be able to route vehicles more quickly and efficiently than a human dispatcher to be of any value. Situations in which automated vehicle routing may be applicable can be characterized as follows:

1. A large number of customers;
2. Many routes with several customers per route;
3. The delivery operation can be standardized in terms of a unit of demand and capacity and in terms of time

standards to perform specific actions such as loading and unloading;

4. Customer volume of demand changes significantly from delivery period to delivery period so as to make fixed schedules impractical;

5. Everyone in the delivery system is interested in being efficient.

Under these conditions it is likely that a computer program can route vehicles more efficiently than a human dispatcher.

In Oakland, customer demand does not vary much from day to day. Therefore, vehicle routes change little from day to day. Furthermore, even though the total number of different customers BALD services over a year is quite large, the number of routes and the number of customers per route each day is small. Also, it is difficult to predict how long it takes to service a particular customer. In one dirver's log it was noted that it took four hours for the truck to be unloaded. In talking to BALD and PWC personnel, it was found that such delays are not uncommon. These delays not only contribute to the cost of delivery, but also make automated vehicle routing very difficult.

Tightening up of current operations seems to be much more important at Oakland than trying to use an algorithm to schedule vehicles. Once this is done and adequate data is obtained, the transportation assets required to make deliveries can be planned using a simple manual procedure such as that suggested by Doll.

The author did not have enough time to study the San Diego local delivery operation and recommend for, or against, automated vehicle routing. However, the IBM VSPX package has been used there in the past with some success. The major problem encountered in its use was that the daily collection of customer demand data was considered to be too time consuming since it was not automated. With the planned automation under the Naval Integrated Storage, Tracking and Retrieval System (NISTARS) concept, that difficulty should be mitigated. If, in addition, the various times needed by a shceduling algorithm can be reliably obtained, then a scheduling algorithm is crucial to its success. The VSPX package used in the past at San Diego was not interactive and manual adjustments had to be made to accommodate customer demands occurring late in the day prior to delivery.

If an algorithm is deemed worthwhile to the local delivery system, then the author would recommend the two-phase algorithm refined by 2-optimal procedures. This algorithm is applicable to a broad range of problems and yields solutions with relatively low objectives in a small amount of computation time.

APPENDIX A

THE TRAVELLING SALESMAN PROBLEM

In this appendix, the travelling salesman problem (TSP) and some of its characteristics are briefly discussed. There is such a tremendous volume of literature on the TSP that it is impossible to review everything written on the subject. Therefore, only two algorithms--one exact and one heuristic--are explained.

A. THE TRAVELLING SALESMAN PROBLEM

The basic TSP involves routing a salesman from his home city through n-1 remaining cities and then returning home so that the total distance (or time or cost) of the tour through the cities is a minimum. It is assumed here that a complete matrix of distances between each pair of cities is given, that the matrix is symmetric and that it satisfies the triangle inequality. Symmetry implies that the distance between any pair of cities is the same, regardless of the direction of travel (i.e., D(i,j) = D(j,i) for any pair of cities P(i) and P(j)). The triangle inequality requires that the direct distance between two cities is shorter (cheaper) than the distance between the same two cities where an intermediate city is visited (i.e., D(i,j) is less than or equal to D(i,k) +D(k,j) for every P(i), P(k) and P(j)). In the general TSP, neither of these assumptions are required. However, for the purposes of this thesis, these assumptions are reasonable.

In general, the TSP is computationally complex. There are (n-1)!/2 possible tours for a problem of n cities. Thus, for a problem of only 10 cities, there are 181,440 possible tours, among which the optimum tour(s) is (are) to be found.

It can be shown that for the conditions stated above, the optimal tour visits each city exactly once. Furthermore, the optimal tour does not intersect itself. This last fact about optimum TSP tour is the motivation for the heuristic algorithm to be discussed.

B. A BRANCH AND BOUND ALGORITHM

Little et al., [Ref. 18] have developed a depth-first branch and bound algorithm for solving the TSP. To clarify the explanation of the algorithm, some definitions are given. An arc (i,j) is the shortest path between two arbitrary cities P(i) and P(j). A tour is a set of arcs that form a single path (called a tour) through all cities and such that exactly two arcs have end points at each city. A node of the search tree in the branch and bound algorithm consists of a set of tours. The term "best solution" is used to mean the shortest tour. Finally, the incumbent solution is the best solution found at any stage of the search.

The concept of the algorithm is to divide the set of tours of a node into two mutually exclusive sets of tours, each of which forms a new node (see Figure A-1). In Figure A-1, node A is the set of all possible tours. This node

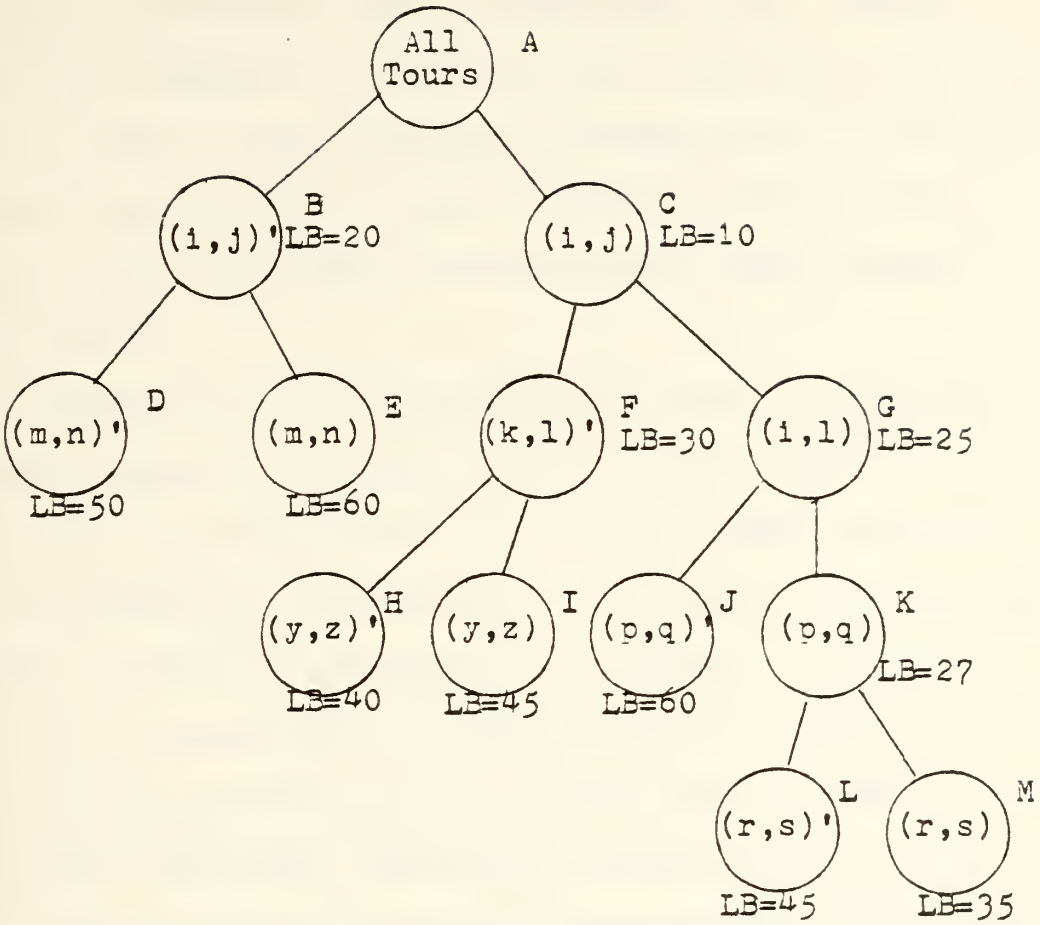

All Tours
A
(i,j)'
B
LB=20
(i,j)
C
LB=10
(m,n)'
D
LB=50
(m,n)
E
LB=60
(k,l)'
F
LB=30
(i,l)
G
LB=25
(y,z)'
H
LB=40
(y,z)
I
LB=45
(p,q)'
J
LB=60
(p,q)
K
LB=27
(r,s)'
L
LB=45
(r,s)
M
LB=35

FIGURE A-1

is divided into two mutually exclusive sets by picking an arc (i,j). Node B (labeled (i,j)') consists of the set of all possible tours that do not include arc (i,j). Node C (labeled (i,j) consists of all the tours that do include the arc (i,j). Next, a lower bound is a number that is guaranteed to be less than or equal to the distance of the shortest tour in that set. A method for determining lower bounds is discussed below.

Many different decision rules may be used to determine which node to branch from next. The rule suggested by Little et al., is to branch from the node with the smallest lower bound. In Figure A-1, since node C has a smaller lower bound than node B, node C is selected to be branched from next. The arc (k,l) is selected to divide node C resulting in node F with a lower bound (LB) equal to 30 and node G with LB = 25. At this point, the nodes available for further division are nodes B, F and G. Since node B has the smallest lower bound, branching continues from that node.

If the branching process is continued far enough, eventually a tour is found. This tour becomes the incumbent solution and the algorithm continues looking for better solutions. If a node has a lower bound that is greater than the incumbent solution, then the optimum solution is known not to be developed beyond that node. Therefore, no further branching from that node need be made. Notice that at node M in Figure A-1, a tour is found with tour length LB equal to 35. Since this tour length is less than the lower bounds

of the remaining available nodes, the optimal tour is found.

Two points now need elaboration. First, a rule for selecting the arc (i,j) at each stage is needed and secondly, a quick method for determining a lower bound on the best tour in each set is required. The rule used for selecting the arc to branch on should divide the current set of tours into two sets--one set where the optimal solution is likely to occur and the other one where the optimal solution is unlikely to occur. Such a rule minimizes the number of branches that must be searched.

Before discussing either of these rules, a discussion of how to reduce the distance matrix is needed (see Table A-I). A matrix is reduced when it is both row and column reduced. A distance matrix is column (row) reduced by subtracting the smallest number in each column (row) from every other number in that column (row). When this is done, every column (row) has at least one zero in it. In Table A-I(a) through A-I(c), an example of a column reduced and a column and row reduced matrix is given. The numbers in column 1 of Table A-I(b) are obtained by subtracting 5 from every number in column 1 of Table A-I(a). The remaining columns of Table A-I(b) are obtained in a similar fashion. The matrix in Table A-I(c) is obtained by row reducing the matrix of Table A-I(b). The reader should ignore the circled numbers for the moment.

The interesting point about adding or subtracting any number from every element of any column (row) is that it does

TABLE A-I

	1	2	3	4	5	6
1	∞	10	5	11	8	6
2	10	∞	7	21	4	15
3	5	7	∞	14	3	17
4	11	21	14	∞	7	6
5	8	4	3	7	∞	13
6	6	15	17	6	13	∞

a

ORIGINAL MATRIX

	1	2	3	4	5	6
1	∞	6	2	5	5	0
2	5	∞	4	15	1	9
3	0	3	∞	8	0	11
4	6	17	11	∞	4	0
5	3	0	0	1	∞	7
6	1	15	14	0	10	∞
	5	4	3	6	3	6

b

COLUMN REDUCED

	1	2	3	4	5	6
1	∞	6	2	5	5	0
2	4	∞	3	14	③ 0	8
3	① 0	3	∞	8	⓪ 0	11
4	6	17	11	∞	4	④ 0
5	3	③ 0	③ 0	1	∞	7
6	1	11	14	② 0	10	∞

c

COLUMN AND ROW REDUCED

not change the optimum solution to the problem. This is true since each city must have exactly one arc leading into (out of) the city. Therefore, changing the length of every possible arc leading into (out of) a city does not change the relative effect of having any particular arc in a tour. Furthermore, the length of a tour after the ditance matrix is reduced differs from the original problem by the sum of the numbers used to reduce the matrix.

To illustrate matrix reduction, consider the tour P(1)-P(2)-P(3)-P(4)-P(5)-P(6)-P(1). From Table A-I(a), the length of this tour is 57. Using the column reduced matrix of Table A-I(b), the tour length is 30. The sum of the reducing constants is 27. Notice that 30 + 27 = 57 and is the length of the tour using the unreduced matrix.

The problem now is to select the arc (i,j) such that if this arc is omitted from the set of possible arcs, the permissible tours are least likely to contain the optimal solution. If, for example, arc (a,b) is omitted, then node P(a) must have an arc leaving it not going to P(b) and P(b) must have an arc coming into it but not from P(a). Thus, the length of the best tour not containing arc (a,b) must be at least as large as the sum of the smallest arc leaving P(a) not going to P(b) and the smallest arc entering P(b) not coming from P(a). Call this distance THETA(a,b). The arc (i,j) selected for omission at any stage is the arc (i,j) such that THETA(i,j) is a maximum over all arcs in the current node. Only arcs (i,j) that have an inter-node distance

D(i,j) of zero in the reduced matrix need be considered. Otherwise, THETA(i,j) is equal to zero. In Table A-I(c), the circled numbers represent the values of the THETA(i,j)'s. As an example, consider the circled number 3 in row 2 and column 5 of the matrix in Table A-I(c). In row 2, the smallest number not in column 5 is 3. In column 5, the smallest number not in row 2 is zero. Thus, THETA(2,5) = 3 + 0 = 3.

Next a method is needed to determine a lower bound on the best solution in the current node. Such a bound can be found by taking the sum of the reducing constants. As described earlier, if a tour found under a non-reduced matrix has length z(t) and the same tour under the reduced matrix has length zl(t), then z(t) = zl(t)+h where h is the sum of the reducing constants. Since the method of reducing the matrix insures that zl(t) $\geq$ 0, it must be true that h $\leq$ z(t) for any tour in the current node. Thus h is a lower bound on the length of the best tour in the current node. A lower bound on the optimal tour for the matrix in Table A-I(a) is 28.

The algorithm now works according to the following steps.

Step 1 initializes the algorithm by setting up the original distance matrix, C, setting the current node, X, equal to the set of all tours and setting the length of the incumbent tour to infinity (since no tours have been found yet).

Step 2 reduces the current distance matrix and labels the current node with its lower bound.

Step 3 selects the arc (i,j) on which to base the next branching. This selection is made as described previously.

Step 4 extends the search tree from the current node, X, to the node Y', in which the arc (i,j) determined in step three is omitted. Node Y' is labeled with its lower bound where LB(Y') = LB(X) + THETA(i,j).

Step 5 sets up the node Y that must contain the arc (i,j). Since arc (i,j) is committed to every tour in node Y, row i and column j of the distance matrix C are no longer needed in node Y and are delted from the matrix. In node Y, there is a set of arcs besides (i,j) that are committed to every tour in node Y. Thus, it is possible that the arc (i,j) is connected to other arcs forming a path leading from some node, P(d), and ending at some node, P(e). Each such arc (d,e) must not be permitted in node Y, otherwise a subtour (or a tour of less than n cities) is formed. To prevent this, each D(d,e) is set to infinity. Now the matrix C of node Y is reduced. The lower bound for node Y is determined to be LB(&) = LB(X) + h where h is the sum of the reducing constants.

Step 6 checks to see if a single tour node is near. If the matrix C has been reduced to a 2X2 matrix, the LB(Y) is equal to the length of the only remaining tour in node Y. If C is 2X2 and LB(Y) is less than the length of the incumbent tour, ZB, the tour in node Y becomes the incumbent, and the length of ZB is set equal to LB(Y). If C is not 2X2 or LB(Y) > ZB, go immediately to step seven.

In step 7 the next node, X, for branching is selected. As mentioned earlier, the node with the smallest lower bound is selected.

Step 8 checks to see if the optimum solution has been found. If ZB $\leq$ LB(X) then the best possible solution has been found and the algorithm terminates.

Step 9 sets up the distance matrix of node X selected in Step 7. If the new node X is equal to the old node Y, the C matrix is already set up and a return is made to Step 3. Otherwise:

1. Set C equal to the original matrix.
2. For each arc (i,j) required to be in all solutions developed from node X, let g = the sum of the lengths of these arcs.
3. For each arc (i,j) committed in node X delete row i and column j. For each path starting at some node P(d) and ending at some node P(e) among the committed arcs, set D(d,e) to infinity. For each arc (r,s) prohibited from node X, set D(r,s) to infinity.
4. Reduce the C matrix.
5. Label X with LB(X) = g + n.

Return to Step 3.

A four-node problem is worked out in Figure A-2, where each node is labeled in capital letters and each matrix is labeled in arabic numerals. The original inter-distance matrix is labeled (1). When matrix (1) is column and row reduced, the result is matrix (2). The sum of the reducing

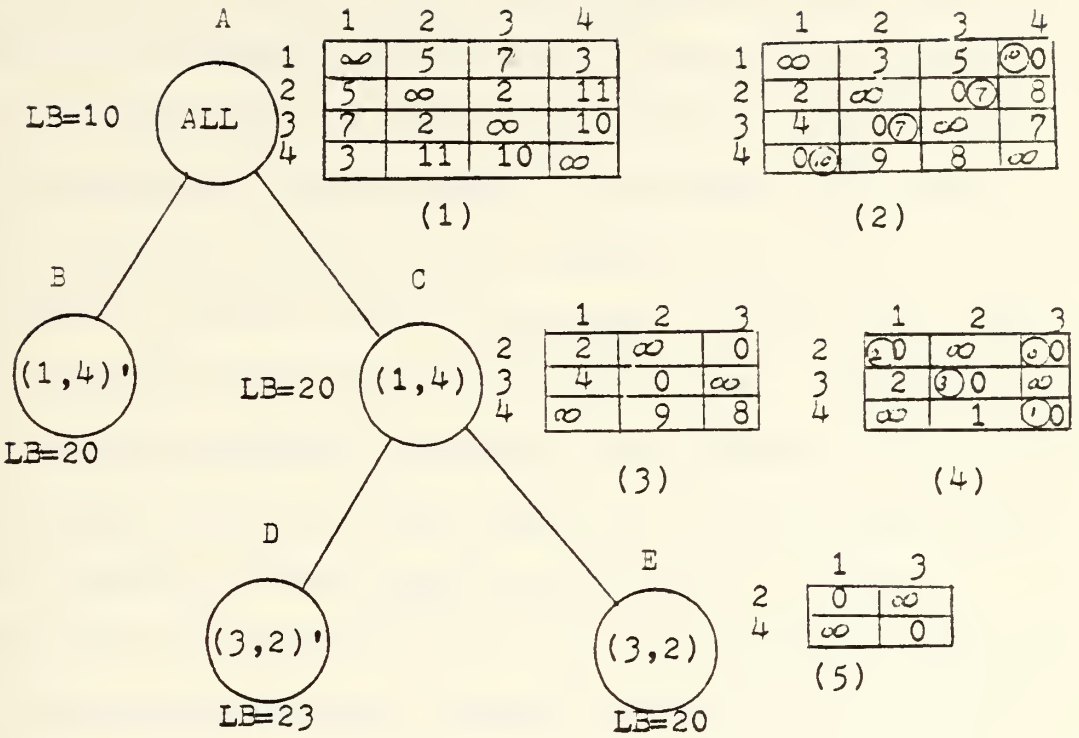


FIGURE A-2

constants is 10 and therefore, node A is labeled with a lower bound of 10. In matrix (2), the circled numbers are the THETA(i,j)'s. Notice that the THETA(i,j)'s are non-zero only where the reduced matrix entry is zero. The largest THETA(i,j) corresponds to arc (1,4). Thus, in node B, arc (1,4) is prohibited and in node C is required. The lower bound of node B is equal to LB(A) + THETA(1,4) = 20.

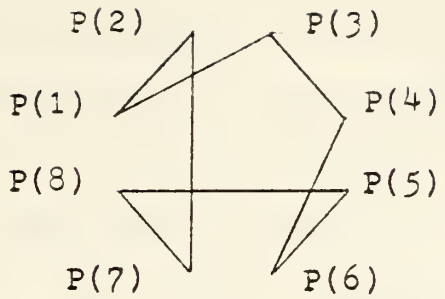
In node C, the arc (1,4) is required. Thus, row 1 and column 4 of matrix (2) are no longer required. Furthermore, D(4,1) is set to infinity to prevent the subtour P(1)-P(4)-P(1). In the reduced matrix (4), the THETA(i,j)'s have been calculated and the sum of the reducing constants, h, is 10. Therefore LB(C) = LB(A) + h = 20. Since both node B and node C have the same lower bound, either node can be selected to branch from. In Figure A-2 node C is selected and the branching process continues. In node E, the reduced matrix is 2X2. The arcs (1,2) and (4,3) of the reduced matrix have 0 as their distances and are therefore the final two arcs required to make a tour along that branch. Node E corresponds to tour P(1)-P(4)-P(3)-P(2)-P(1) with length 20. Since node E has a lower bound that is less than or equal to the lower bound of any remaining node, an optimum solution has been found.

C. THE R-OPTIMAL HEURISTIC

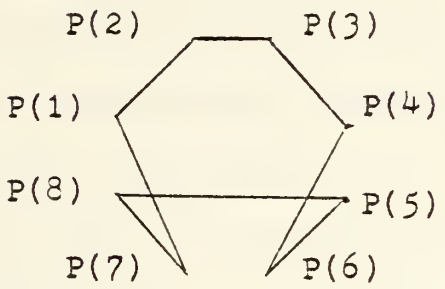
The r-optimal heuristic, developed by Lin [Ref. 8: p. 132-135], is an improvement algorithm which replaces r links of an incumbent tour with r new links that result in a tour and

such that the new tour is shorter in length. The new tour then becomes the incumbent tour and the process of replacing r links continues. The algorithm terminates when no improvement can be found by replacing r links of the incumbent tour with r new links. A tour which cannot be improved by replacing r links is called r-optimal. Before this algorithm can be used, an initial tour must be generated. In general, different initial tours result in different r-optimal tours. Therefore, a strategy that can be employed is to generate several initial tours, find the r-optimal tour resulting from each and then select the shortest tour. In order to prove that a tour is r-optimal, every combination of r links must be examined for possible replacement. Since there are $C = n!/(r!(n-r)!)$ ways of choosing r arcs from n arcs, a problem with n cities requires C checks be made to prove that a tour is r-optimal.

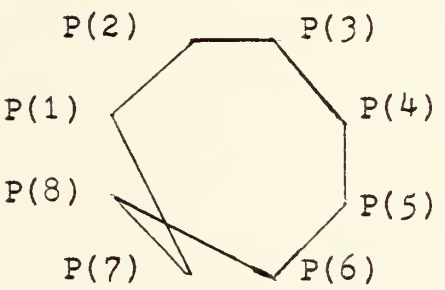
As an example, consider the problem in Figure A-3 where each figure is drawn to scale. Starting with the initial tour in Figure A-3(a) and using r = 2, consider removing arc (1,3) and arc (2,7). When these two arcs are removed, the only possible replacements that maintain a tour are arc (2,3) and arc (1,7). Since the new tour is better than the old tour, the new replaces the old and the process continues. The algorithm terminates when no two arcs can be replaced by two other arcs that result in an improvement. In this problem, there are eight arcs that make up a tour. Since there are 28 ways to select 2 arcs from 8 arcs, the algorithm terminates



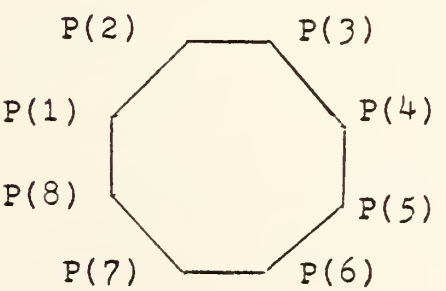
(a)



(b)



(c)



(d)

FIGURE A-3

after checking 28 pairs of arcs. An interesting property of two-optimal tours is that they do not intersect themselves. Notice that the tour of Figure A-3(d) is intersectionless.

When r is greater than 2, there is more than one way to form a new tour when r arcs are removed. Specifically, when r = 3, there are eight ways of replacing 3 arcs and still maintaining a tour. This fact is demonstrated in Figure A-4. In Figure A-4(a), the arcs P(1)-P(8), P(3)-P(4) and P(5)-P(6) are removed from the tour P(1)-P(2)-P(3)-P(4)-P(5)-P(6)-P(7)-P(8)-P(1). The seven additional ways of forming a tour are shown in Figures A-4(b) through A-4(h).

Obviously a tour that is n-optimal is the optimal tour. However, this algorithm has little practical use beyond r = 3 because the computation time becomes too large.

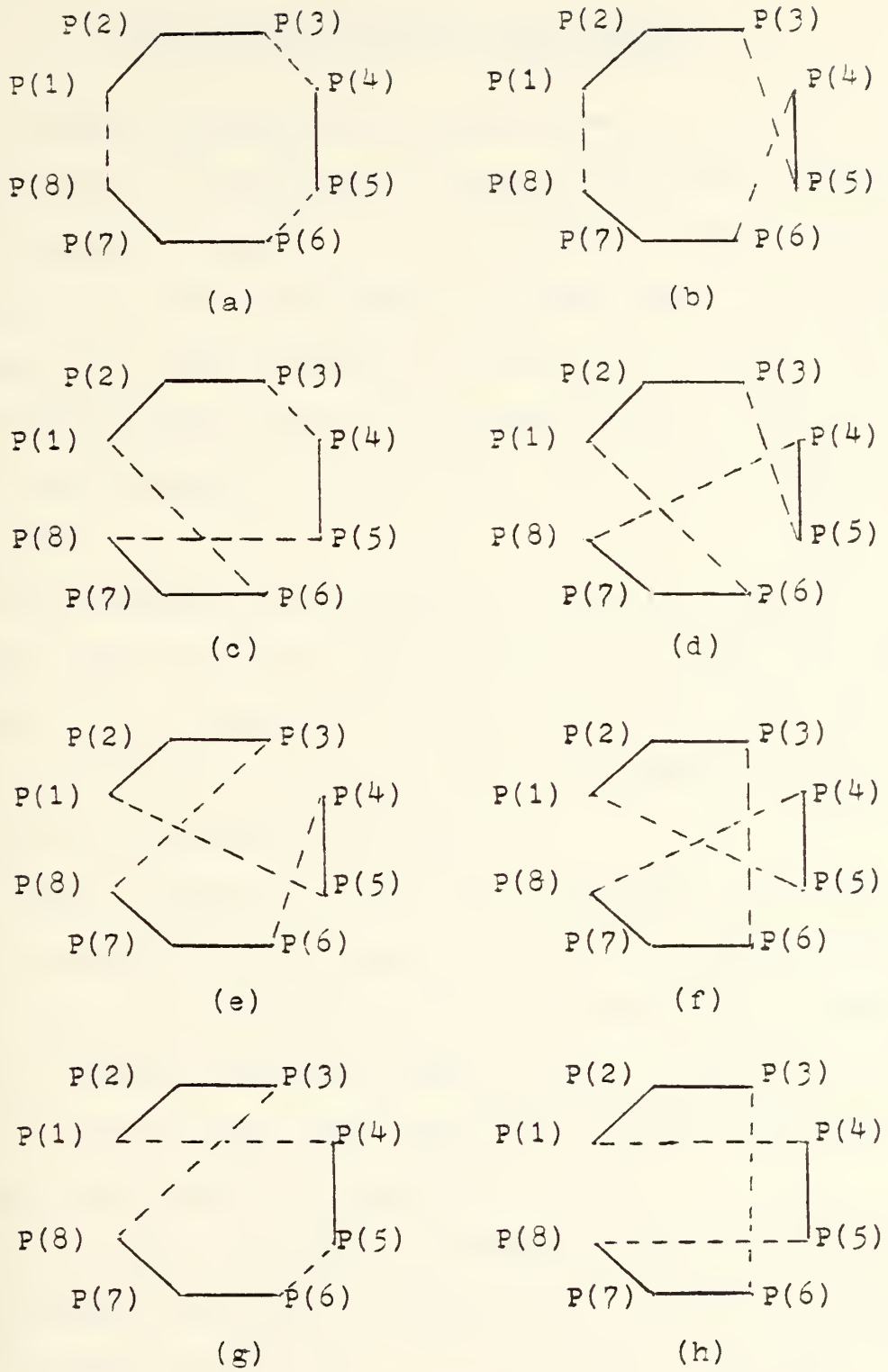


FIGURE A-4

APPENDIX B

THE MINIMUM SPANNING TREE PROBLEM

The minimum spanning tree problem [Ref. 13: p. 220-224] can most easily be explained in terms of providing telephone lines to each of n cities. It is desired to provide these lines such that the total length of cable used is a minimum and there is a line traceable from any city to any other city. The minimum spanning tree can be found quickly and also provides a lower bound in the heuristic tree search algorithm for the VRP.

The algorithm is initialized by selecting a city, P(i). The first step is to find the city that is closest to city P(i), say P(j). Connect these two cities with arc (i,j). Cities P(i) and P(j) now form a set of connected cities. Call this set of connected cities, S.

In Step 2, find the cities not in the set S that are closest to each of the cities in the set S. Connect the two cities (one in S and one not in S) corresponding to the smallest distance and add the newly connected city to the set S. If there are unconnected cities remaining, repeat this step. Otherwise, the algorithm terminates.

As an example, consider a problem for which the intercity distances are given in Table B-I. City P(1) is selected to initialize the algorithm. The city P(4) is closest to P(1). Therefore the arc (1,4) is added to the tree first.

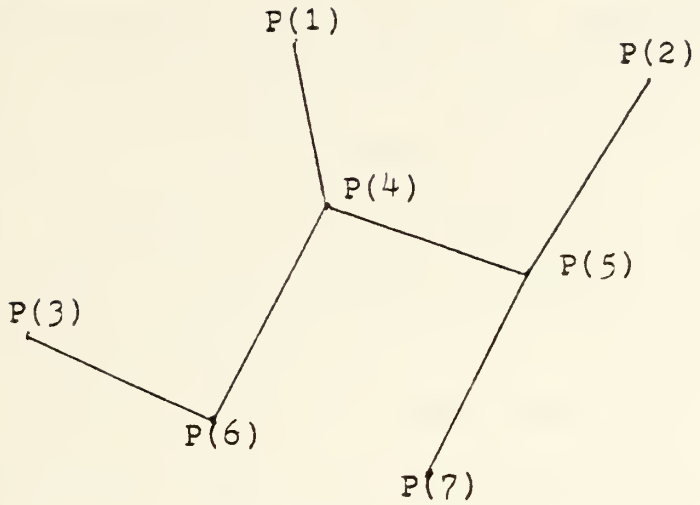


FIGURE B-1

TABLE B-I

P(1)	P(2)	P(3)	P(4)	P(5)	P(6)	P(7)
20						
10	30					
5	12	11				
17	7	25	8			
15	25	9	7	12		
23	21	24	11	7	9	

At this point the cities P(1) and P(4) are connected and are therefore put into the set S. The next city closest to P(1) is P(3) and the next city closest to P(4) is P(6). The distance between P(4) and P(6) is 7 which is less than 10, the distance between P(1) and P(3). Therefore the arc (4,6) is added to the tree and P(6) is put into the set. The next city closest to a city in the set S is P(5). Therefore the arc (4,5) is added to the tree and P(5) is put into S. The final solution to the problem is shown in Figure B-1. The sum of the arc lengths is 43 which is a lower bound on the total distance of a set of routes that visit the customers P(1) through P(6) (see the tree search algorithm).

APPENDIX C

THE SHORTEST PATH PROBLEM

The shortest path problem [Ref. 6: p. 217-220] concerns finding the shortest path from an origin to a destination through a connected network. If the network is thought of as a system of streets where the nodes are street intersections and the arcs are the streets connecting each intersection, then the problem is equivalent to finding the shortest route from any street intersection to any other street intersection.

In the discussion that follows, the term directly connected is used. Two nodes P(i) and P(j) are said to be directly connected if the arc P(i)-P(j) exists. Before the algorithm can be applied, a network with an origin node and a destination node must be given and the length of each arc in the network must be known. A simple network is shown in Figure C-1. The origin node is labeled P(0), the destination node P(7) and each arc is labeled with its length.

The first step of the algorithm is to find the node P(i) closest to the origin. Node P(i) and P(0) are then placed in the set of nodes, S, for which the distance from the origin is known.

In the second step, the algorithm looks for the node, P(j), not an element of S that is closest to the origin. The node P(j) must be directly connected to one of the nodes in the set S. If P(j) is not directly connected to a node in

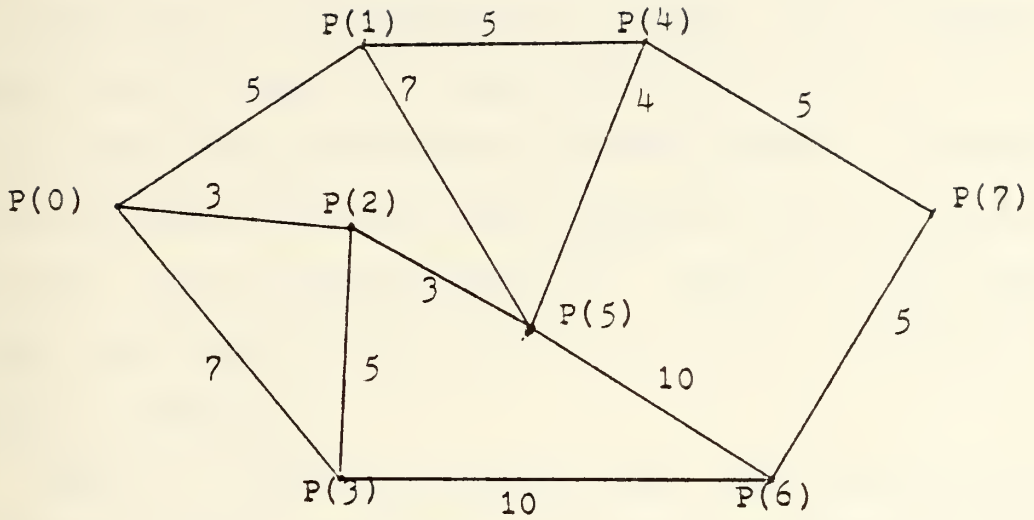


FIGURE C-1

TABLE C-I

Stage	P(j)	Connection	D(0,j)
1	P(2)	P(0)	3
2	P(1)	P(0)	5
3	P(5)	P(2)	6
4	P(3)	P(0)	7
5	P(4)	P(1),P(5)	10
6	P(7)	P(4)	15

S, then there exists a node P(k) not an element of S that is closer to the origin. If P(j) is the next node to enter S and P(j) is directly connected to P(i), an element of S, then the shortest distance between P(0) and P(j), D(0,j), is D(0,i) + D(i,j). The distance D(0,j) and the node P(i) to which P(j) is connected are recorded. It is possible that more than one node in S, when connected to P(j), results in P(j) being the same distance from the origin. Therefore, each of these nodes are also recorded as being connected to P(j). P(j) is then added to S. If P(j) is the destination node, the algorithm continues on to Step 3. Otherwise, this step is repeated.

Step 3 is the backtracking step. When the destination node is reached, each chain of arcs terminating at the destination node is traced back to the origin. Each such chain forms a shortest path between the origin node and the destination node.

As an example of this algorithm, consider the problem illustrated in Figure C-1 in which P(0) is the origin and P(7) the destination. The solution is completely worked out in Table C-I. At stage 1, P(2) enters into S because P(2) is closest to the origin. The node P(2) is connected to P(0) and is a distance of 3 from the origin. At stage 2, P(1) enters S because D(0,1) = 5 is less than D(0,2) + D(0,5) = 6 (where P(r) is the node closest to P(2)). This process is continued until node P(7) is reached. Notice that P(4) is

connected to both P(1) and P(5) because D(0,1) + D(1,4) = D(0,5) + D(5,4) = 10.

Next the backtracking process is used to determine the shortest path(s) between P(0) and P(7). From Table C-I, P(7) is connected to P(4). Therefore, an arc of the shortest path is P(4)-P(7). The node P(4) is connected to both P(1) and P(5). Therefore, two separate paths back to the origin exist, one through P(1) and the other through P(5). Continuing the backtracking procedure yields paths P(7)-P(4)-P(1)-P(0) and P(7)-P(4)-P(5)-P(2)-P(0), both of which have length 15.

APPENDIX D

AUTOMATED VEHICLE SCHEDULING PROGRAMS

A. THE IBM VSPX

The IBM Vehicle Scheduling Program-Extended (VSPX) is designed specifically for the IBM system/360 and IBM system/370. In its current form this program dates back to 1970. It routes vehicles from a central depot to service the demand of a set of customers so as to minimize total route distance or time. Its method is based on the algorithm of Clark and Wright and includes many day-to-day operational conditions in producing routes. These conditions may be segregated into customer and vehicle conditions. Permissible customer conditions are as follows:

1. Time windows for each customer may be specified.
2. An average fixed time per stop in addition to time required for loading and unloading may be set.
3. Special time involved in making deliveries to specific customers can be specified.
4. Vehicle size or type restrictions by customer are permissible.

In addition to these customer conditions, the following fleet or route conditions can also be included:

1. Up to 255 different vehicle types may be used.
2. The average fleet speed may be modified to account for special conditions such as weather.
3. The earliest start time and the latest finish time for the whole fleet may be specified.

4. The maximum permissible route time by vehicle type can be specified.

5. The maximum number of stops per route can be set.

6. Vehicles may have up to fifteen different compartments specified by vehicle type. Compartments allow separation of quantities by customer.

7. Up to fifteen vehicle and trailer types can be designated along with information as to which vehicle type can be connected to which trailer type.

8. The average unload time per unit of material can be specified. This time is used in calculating customer stop time.

9. Two dimensions of quantity (e.g., weight and cube) can be used in describing demand or capacity.

10. More than one route can be serviced by a vehicle during a delivery period.

11. Journeys lasting more than one day can be handled.

12. Low priority loads can be specified. This option gives the user the ability to quickly delete low priority deliveries from the delivery schedule if a schedule turns out to be infeasible.

13. This system uses the zone concept described in Chapter IV. One average intercustomer time (distance) within a zone applicable to all zones can be specified to help in calculating route time.

14. A maximum route mileage can be specified.

15. A predetermined begin or end point of a route can be selected.

Two options for entering customer inter-stop distances are available. The two options are exact distance and euclidian distance, both of which are described in Chapter IV. Depending on the capabilities of the particular IBM system used, VSPX can schedule anywhere from 250 to 1650 customers per run.

This program is not interactive. Once the required data is collected and input, the program is run. Any changes to the resulting routes must either be made by manually editing the output or by changing the input data and running the program again.

B. IVESS

McDonnel Douglas Automation Company (MCAUTO) markets a system called Interactive Vehicle Scheduling System (IVESS) developed by Decision Graphics. This system solves the same problem as the IBM system. It has generally the same capabilities as the IBM system, but, unlike VSPX, is interactive. All day-to-day delivery requirements are input through a remote terminal to a computer in St. Louis. The computer-generated routes and a map of the delivery area along with customer information can be displayed at the terminal. Then, any route infeasibilities that either the computer does not recognize or cannot resolve can be taken care of by the human operator through a set of simple commands. These commands allow movement of customers between and within routes.

The system includes several different heuristics that generate the computer solutions. The heuristic used depends on the specifics of the particular dispatch operation. Interface with the computer can be made with most Tektronix graphics terminals which can be bought or rented from MCAUTO. It is also possible to use terminals at local MCAUTO offices. Charges for the use of the system include any labor involved in setting

up or modifying data and computer time. Charges for computer time vary dependent on the type of job but average about $75 per clock hour. The system is capable of scheduling up to 187 customers per run.

C. AVS

The Automatic Vehicle Scheduling (AVS) program developed by David W. Taylor Naval Ship Research and Development Center is an interactive program designed specifically for the routing of vehicles at the NSC in Charleston, N.C. The dispatching problem that AVS addresses is different than the problem IVESS and VSPX are designed to handle. At Charleston there are 92 potential pick-up and delivery sites, any one of which may be making deliveries to or receiving deliveries from any of the other 92 sites. Additionally, of the 92 sites, only six are located remote from the center. The AVS system does not combine off-center locations into routes. In other words, any trucks going to off-center locations are scheduled to leave an on-center location, visit a single off-center location and return to the base.

All vehicles are dispatched from a single building to service a set of orders, where an order consists of a pick-up and a delivery. The program combines orders into a set of routes that are both time-feasible and capacity-feasible. The AVS system can handle up to 99 pick-up and delivery sites and four vehicle types. The program is written in FORTRAN and is designed specifically for the Burroughs B3500 computer.

D. CONCLUSION [Ref. 2]

In addition to the aforementioned programs, many other specialized systems are available commercially, some with true optimization capabilities. The efficiency and cost of these systems varies between jobs. Generally, good performance for local delivery operations requires interactive access, a customer-tailored algorithm, a well understood set of instructions for the human operator, and real-time computer resources which commonly cost as much as operating several vehicles. Experience with these systems has revealed that "thumb-rules" are not always reliable, that significant operational savings can be achieved, but that the managerial context must admit necessary discipline. This discipline is justified and accepted primarily in terms of reducing operating costs. For Navy operations, it may be difficult to achieve such a managerial environment, and thus to succeed at automated vehicle scheduling.

LIST OF REFERENCES

1. Beltrami, E.J. and Bodin, L.D., "Networks and Vehicle Routing for Municipal Waste Collectors," Networks, v. 4, p. 65-94, 1974.

2. Brown, G. and Graves, B., "Real-Time Dispatch of Petroleum Tank Trucks," Management Science, v. 27, p. 18-31, Jan 1981.

3. Christophides, N. and others, ed., Combinatorial Optimization, John Wiley and Sons, 1979.

4. Clark, G. and Wright, J.W., "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," Operations Research, v. 11, p. 568-581, 1964.

5. Doll, L.C., "Quick and Dirty Vehicle Routing Procedure," Interfaces, v. 10, p. 84-85, February 1980.

6. Dreyfus, S.E., "An Appraisal of Some Shortest Path Algorithms," Operations Research, v. 17, p. 395-412, 1969.

7. Eilon, S., and Christofides, N., "An Algorithm for the Vehicle Dispatching Problem," Operational Research Quarterly, v. 20, p. 309-318, 1969.

8. Eilon, S., Watson-Gandy, C.O.T. and Christophides, N., Distribution Management: Mathematical Modelling and Practical Analysis, Charles Griffin and Company, 1971.

9. Gaskell, T.J., "Bases for Vehicle Fleet Scheduling," Operational Research Quarterly, v. 18, p. 281-295, 1967.

10. Gillett, B.C. and Miller, C.A., "A Heuristic Algorithm for the Vehicle Dispatch Problem," Operations Research, v. 22, p. 340-349, 1974.

11. Golden, B.L., Magnanti,T.L. and Nguyen, H.Q., "Implementing Vehicle Routing Algorithms," Networks, v. 1, p. 113-148, 1977.

12. Hallberg, M.C. and Kriebel, W.R., "A Routing Algorithm Using The Nearest Neighbor Concept," American Journal of Agriculture Economics, v. 61, p. 87-90, February 1979.

13. Hillier, F.S. and Lieberman, G.J., Operations Research, Holden-Day Inc., 1974.

14. Holmes, R.A. and Parker, R.G., "A Venicle Scheduling Procedure Based Upon Savings and a Solution Perturbation Scheme," Operational Research Quarterly, v. 27, p. 83-92, 1976.

15. Hrabosky, B., Owen, W.A. and Popp, G.R., Pre-Consolidation Supply Support for NARF Alameda and NSC Oakland Local Customers, Masters Thesis, Naval Postgraduate School, 1980.

16. IBM Corp., IBM Vehicle Scheduling Program Extended (OS and DOS) General Information Manual, Report GH 19-2000/0, White Plains, N.Y. 1970.

17. Levy, L., Golden, B. and Assad, A., "The Fleet Size and Mix Vehicle Routing Problem," Working Paper MS/S 80-011, University of Maryland, 1980.

18. Little, J. and others, "An Algorithm for the Travelling Salesman Problem," Operations Research, v. 11, p. 498-516, 1963.

19. Mole, R.H., "A Survey of Local Delivery Vehicle Routing Methodology," Journal of the Operational Research Society, v. 30, p. 245-252, 1979.

20. Mole, R.H. and Jameson, S.R., "A Sequential Route-building Algorithm Employing a Generalized Savings Criterion," Operational Research Quarterly, v. 27, p. 503-511, 1976.

21. Navy Fleet Material Support Office, "Navy Automated Transportation Documentation System Requirements Statement," FMSO Document No. NAVADS RS-01, 28 February 1978.

22. Russel, R.A., "An Effective Heuristic for the M-tour Travelling Salesman Problem with Some Side Conditions," Operations Research, v. 25, p. 517-524, 1977.

23. Schultz, H., "A Practical Method for Vehicle Scheduling," Interfaces, v. 9, p. 13-19, May 1979.

24. Tillman, F.A. and Cochran, H., "A Heuristic Approach for Solving the Delivery Problem," The Journal of Industrial Engineering, v. 19, p. 354-358, 1968.

25. Tyagi, M.S., "A Practical Method for Truck Despatching Problem," Journal Operations Research Society of Japan, v. 10, No. 3 and 4, p. 76-92, June 1968.

26. Webb, M.H.J., "Relative Performance of Some Sequential Methods of Planning Multiple Delivery Journeys," Operational Research Quarterly, v. 23, p. 361-371, 1972.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Defense Logistics Study Information Exchange United States Army Logistics Management Center Fort Lee, Virginia 23801	1
3.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
4.	Department Chairman, Code 55 Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
5.	Professor A.W. McMasters, Code 54Mg Naval Postgraduate School Monterey, California 93940	5
6.	Professor G.G. Brown Code 55Bw Naval Postgraduate School Monterey, California 93940	5
7.	CPT Clifford Clausen 9567 Meadow Green Circle Sacramento, California 95827	1
8.	Mr. H.J. Lieberman Code 0431B Naval Supply Systems Command Washington, D.C. 20376	1
9.	Director, Operations Analysis Office Code 08.3 Naval Supply Center Oakland, California 94625	5
10.	LCDR J.R. Bailey, Code 49 Naval Supply Center San Diego San Diego, California 92132	5

11. Commanding Officer, ATTN Code 93 — 1
Navy Fleet Material Support Office
Mechanicsburg, Pennsylvania 17055

12. CPT Kris Drach — 1
Code 41.2
Naval Supply Center
Oakland, California 94625

192327
Thesis
C503 Clausen
c.1 Vehicle routing algorithms for local delivery at Naval Supply Centers.